# Introduction

## 1.1. Purpose and scope

The Graduation Project is a culminating exit course in Baccalaureate, Master and Doctorate education. A team member of a graduation project demonstrates an ability to apply knowledge and skills, to solve a problem and to participate in team work activities. He/She will undertake research work under the guidance of a supervisor. He/She defines the project problem, formulates the goals and collects information to achieve the objective. Then he/she writes a technical report. Finally, he/she makes a presentation of his/her project for the discussion committee.

The team members of any graduation projects face many difficulties (in home working) to work together in the most natural manner. Pass by this, we aim to develop a suitable online environment ”CFGP” as face-to-face to help team members geographically distributed communicating on their graduation project.

## 1.2. Contribution: CFGP web based tool

The objective of our tool is to provide an efficient environment that facilitates the communication and makes the work more flexible.

To achieve our goal, we add some features to the tool by gathering and analyzing the requirements from real teams who have many crises in their project. As example, we add discussion rooms, shared folders, tasks schedule and shared board. There are different access rights depending on the type of user: member, coordinator and supervisor. This specification is a benefit of our tool for making the work more organized. The expected outcomes are:

- Provide a framework for each team with special account.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

- The user can see if the other users in his group online in the contact list. It is done by session of the tool that records each member has log in the database.

- There are shared folders: one for team work and the other for dealing with the supervisor. These folders are laid on the server of our tool. Our tool enables any member to upload, download file or groups of files and directories by compress them over server through HTTP protocol.

- Presents an advertising space that the adding is enabled for each user and the others can add comments to it. It is supported by session and accessing database.

- Includes two discussion rooms: one is private for the team members only and the other is open for all by using HTTP protocol. This feature provides a synchronized communication by using acquire release protocol.

- The team members can view the log file which includes the records for many events such as editing and creating files. Log file reserves action description, actor, time and date. It lists only last ten actions. It depends basically on recording into database.

- There is tasks schedule for team member work including distributed work over members and appointment times created by HTTP response of web server to HTTP request posted by coordinator.

- Includes a questionnaire to choose appointment times that appropriate for most members. A system will display if there is a meeting or not before one day of the meeting suggested day according to the percentage of AYES (members vote yes) and percentage of NOES (members vote no).

- Includes synchronous shared board for illustrating members ideas graphically by using java applet, the TCP socket in the TCP layer and synchronized communication by using acquire release protocol.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

- Password’s encryption by using shared key algorithm on server side.

## 1.3. Document Organization

The report is organized as the following. Chapter (2) that gives an overview of cooperative work and how CSCW is applied in distributed system. Then, chapter (3) analyses the CFGP requirements and its intended environment. Then, chapter (4) describes the general architecture of CFGP for both two sides: client and server. Also, it presents the communication model that CFGP tool is based on. The implementation is discussed in chapter (5) which includes our choice of technologies and techniques to build our tool and the logical reasons of this choice. We select java as a programming language and mySQL as a database system. Then, chapter (6) represents the real testing of our tool. Finally, the last chapter gives the conclusion of our work.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

# 2. State of Art

## 2.1. Introduction

A wide range of theoretical and empirical analyses emphasize the importance of active participation and collaboration among people in promoting the effectiveness of online interaction. Cooperative work faced many problems such as communication, cooperation, coordination, solving problems and negotiation. These problems are most common in many cases like academic and business field besides other fields.

Distributed systems have traditionally dealed with these problems that associated with the inter-connection of a number of computer systems. The emergence of Computer Supported Cooperative Working (CSCW) has challenged this technological perspective by focusing on the group-working of a number of users. CSCW allows distant people to interact socially and develops a feeling of community in online team.

This Chapter represents the state of the art in CSCW and relates the general characteristics of existing CSCW systems to a variety of trends in distributed computing. It is argued a reasonable correspondence exists between CSCW and distributed systems in terms of their geographical nature and style of interaction.

## 2.2. CSCW

Computer-Supported Cooperative Work (CSCW) is a software tools and technology to support groups of people working together. CSCW enhances people understanding and may give people a sense of belonging online. Typical applications include email, video conferencing, chat systems, multi-player games and real time shared applications (such as collaborative writing or drawing)[1]. The goal of CSCW is to discover ways of using computer technology to further enhance the social interaction of people in a group work process through support in the time and place dimensions.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

## 2.2.1. The Description of CSCW

The term CSCW referring to the interests of a number of researchers involved in the use of computers to support user groups. The area has evolved to combine the understanding of the nature of group working with the enabling technologies of computer networking, systems support and applications. Five general classes of cooperative system have emerged:

**1- Message Systems**

Message systems are descendants of early electronic mail programs which allowed a user using a central machine to send textual messages to other users on the same machine. As wide area networks designed to support computer communication became more widespread, electronic mail systems increased in complexity and functionality. Each message system makes use of a particular message format to transfer Information (a message format is often defined as a part of most message standards) [2].

**2- Computer Conferencing**

A typical computer conferencing system consists of a number of groups called conferences, each of which has a set of members and a sequence of messages. Conferences are often arranged so that they individually address a single topic. A user subscribes to those conferences which are of interest [3]. The development of reliable high speed communications has lead to the emergence of new real-time conferencing systems. In addition, modern workstations have led to the development of desktop conferencing systems which merge real-time conferencing capabilities with modern windowing environments allowing shared windows across a number of workstations [4].

**3- Meeting Rooms**

A typical automated face to face meeting room consists of a conference room furnished with a large screen projector, a computer (or network of computers), a number of individual input/voting terminals, and a control terminal. The computer system supporting the meeting often makes use of multi-users software. This Software for graphics and vote tally and display also form part of the normal meeting facility [5].

### 4- Co-authoring and Argumentation Systems

Co-Authoring and argumentation systems are a general class of system which aim to support and represent the negotiation and argumentation involved in group working. The cooperative authoring of documents is demonstrative of this class of cooperation where the final generation of a document represents the product of a process of negotiation between authors [6].

### 5- Cooperative Task Management

Task management is a process of managing task during it's lifecycle, including planning, testing, tracking and reporting. Effective task management suppose managing all aspects of a task including its status, priority, time, human and financial resources assignments, recurrency, notifications and so on [7].

One can often divide the work into conceptually separate tasks. Each task encapsulates a control flow, and all of the tasks access some common and shared state. High-performance programs are often written with preemptive task management, where in execution of tasks can overlap on multiprocessors. The opposite approach, serial task management, runs each task to completion before starting the next task. Its advantage is that there is no conflict of access to the shared state (see figure 2.1) [7].

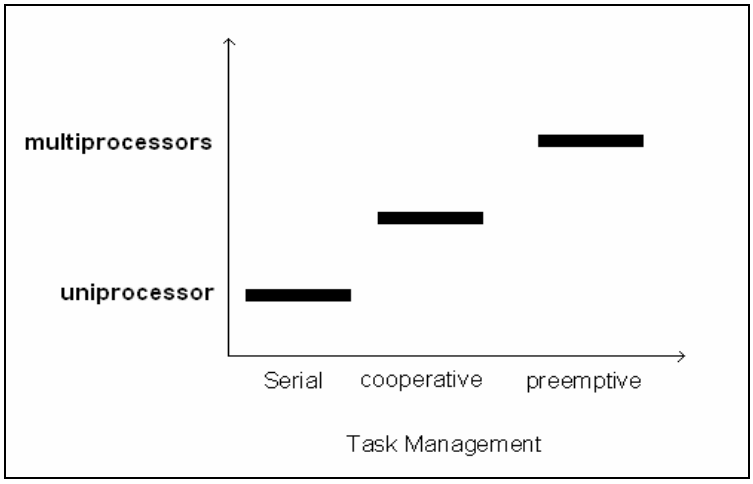



**Figure 2.1: Types of Task Management**

A cooperative task management is a tasks code only yields control to other tasks at well-defined points in its execution. It is valuable when tasks must interleave to avoid waiting on each others, but multiprocessor parallelism is not crucial for good application performance.

Cooperative task management preserves some of the advantage of serial task management in that invariants on the global state only need be restored when a task explicitly yields, and they can be assumed to be valid when the task resumes. Cooperative task management is harder than serial in that, if the task has local state that depends on the global state before yielding, that state may be invalid when the task resumes. The same problem appears in preemptive task management when releasing locks for the duration of a slow operation.

The task manager allows members of distributed teams to share and manipulate common tasks like Windows Task Manager on Microsoft Windows and Activity Monitor on Mac OS X [7].

## 2.2.2. Dimensions of CSCW

The wide variety of CSCW systems highlighted in section 2.2.1 reflects different views of cooperation. The nature of cooperation has been an on- debate

within the CSCW community and two principal characteristics have emerged from this debate [8]:-

i. The form of cooperation.

ii. The geographical nature.

### 1- The Form of Cooperation

CSCW systems are primarily concerned with supporting a number of users cooperating to address a particular problem, or range of problems. The nature of this cooperation can be distinguished by the way in which the group members interact. People can either interact or cooperate synchronously or asynchronously.

Synchronous interaction requires the presence of all cooperating users while asynchronous cooperation occurs over a longer time period and does not require the simultaneous interaction of all users.

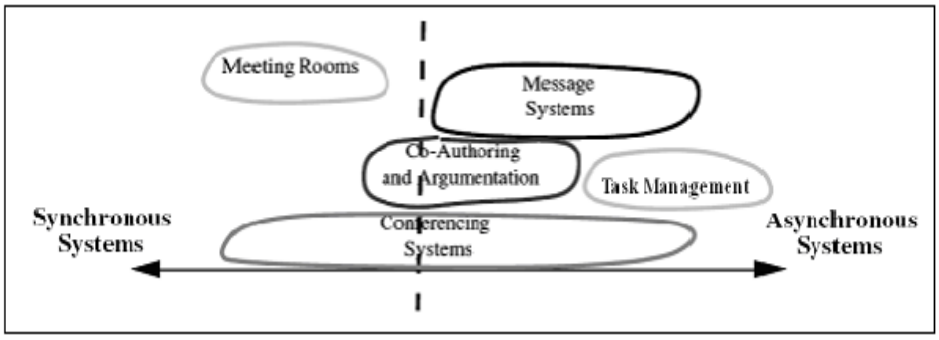



**Figure 2. 2: Form of cooperation in CSCW**

Figure 2.2 shows how the classes of CSCW systems identified in section 2.2.1 fit into this division. From this view, three general classes of CSCW system can be highlighted [9]:

i. **Purely synchronous systems:** Purely synchronous systems need the simultaneous presence of all users. Systems which typify this approach include real-time conferencing systems and meeting rooms.

ii. **Purely asynchronous systems:** Asynchronous systems are designed to facilitate cooperation without the simultaneous presence of all group members. Often the cooperation takes place over a much greater timescale. Cooperative message systems are a primary example of this type of system where users take on independent roles which produce and consume messages. Also, there is another example which is cooperative task management that does not need simultaneous exist of all users. Assume an asynchronous mode of cooperation with users reading and adding articles to a site independently of other users.

iii. **Mixed Systems:** Mixed systems contain elements of support for both synchronous and asynchronous cooperation. They allow real-time synchronous cooperation to take place within the same framework as time-independent asynchronous working. The primary examples of this type of systems are in computer conferencing and co-authoring and argumentation systems.

**2- Geographical Nature**

Cooperative systems are often considered as being either remote or co-located. Co-located systems require the local presence of all users. This five class of system normally takes the form of built meeting rooms which is one of the classes of CSCW mention in section 2.2.1. Remote cooperative systems assume that the accessibility between users is minimal. Remote systems include more sophisticated distributed applications such as multimedia conferencing systems which is on of classes of CSCW mention in section 2.2.1[10].

Some classes of system and how they vary in terms of principle characteristics identified above are shown in figure 2.3.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

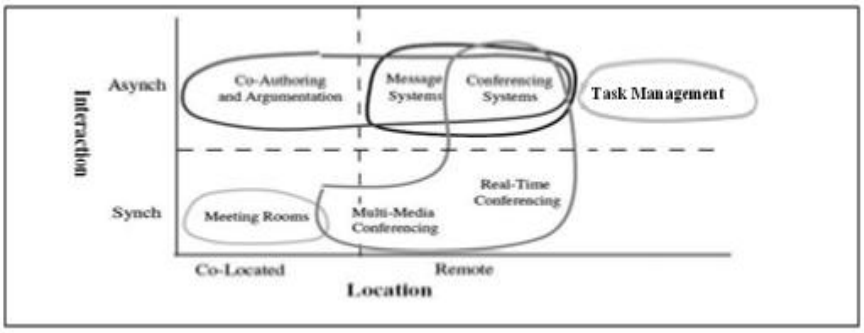



**Figure 2. 3: The Characteristics of CSCW system**

## 2.3. CSCW and Distributed Systems

Distributed systems viewed traditionally in terms of support cooperation between a numbers of computers connected by a network. Note that the term cooperation is used in this context to refer to how closely the computers related to each other within the distributed systems. This interpretation is used throughout this section, rather than the more general application of the term at section 2.2 (CSCW).

### 2.3.1. CSCW Control

CSCW Control is a critical aspect of any distributed application, and ideally should be provided in an obvious way as possible. Consequently, this subsection will illustrate the control of cooperation in CSCW systems.

#### 2.3.1.1. Control Requirements for CSCW

It is difficult to derive precise cooperation requirement to be controlled within CSCW systems. However, any collaborative working environment, particular areas must be addressed. They are as following [14]:

**1- File Management**

File management is essential for a networked collaborative project dealing not only with the overall control of files, but also administration and distribution of files. Controls are required to monitor and administrate downloading and uploading of files; datasets or supporting project documentation. This ensures that the determining the status and location of any file would be at anytime.

**2- Communications**

In a distributed collaborative environment, good communications are essential to ensure that all participants are kept abreast of evolving issues concerning the project. A mechanism must be in place that can provide this good communication. For example, communication security issues associated with access, archiving, and storage is important with a collaborative project. Access rights must exist for participants to view project documentation without permitting unapproved changes.

3- **Security**

Cooperation work as a whole must be protected from outside influences either accidental or malicious in intent. Some of the key areas identified for security protection are communications and file transfer. Special security precautions must be taken in order that the integrity of the controlling CSCW system and its data are protected.

### 2.3.1.2. Examples of Control in CSCW

There are some examples of system based on the formality and nature of control exhibited by each, as a following [18], [19], [20]:

1. **Speech act or conversation based systems**

Cooperation is represented and controlled within this class of system using some form of network structure detailing the patterns of message exchange. For example, instant

messaging, chat rooms and other forms of synchronous computer-mediated communication.

2. **Active Message systems**

Active message systems assume an asynchronous and remote mode of cooperation. The assumption underlying these systems is the members of a group cooperate by exchanging messages. For example, electronic mail system is an active message system.

3. **Conferencing systems**

Conferencing systems provide basic control mechanisms which are minimal and fixed into applications. In more modern real time conferencing systems the control mechanism of the conference imbedded in the conferencing application and controlling who has access to some shared space at any given time.

4. **Peer- group meeting**

Peer meeting systems do not provide any control mechanisms and rely on the meeting participants to formulate their own meeting protocols. All users have equal status and may use the systems freely. The amount of control provided by cooperative systems highlights the level of automation of the system provides. The approaches above can be characterized as either explicit or implicit control. In explicit controlling, users may view group interaction and it is allowing the representation and editing of control information. The first three classes are example of explicit controlling. In contrast, implicit controlling provides no techniques for representing or coordinating group interaction. Implicit approach emerges in the rest of classes.

### 2.3.1.3. Supporting CSCW Control in Distributed Systems

Distributed systems view control as dealing with the problems of distribution and masking such problems from applications (distribution transparency). Distribution transparency is now seen as a collective name for the masking out of

various features of a distributed computation [15]. In effect, there are a number of individual transparencies corresponding to each of these features (see table 2.1).

**Table 21 . The Various Forms of Transparency in distributed System [15]**

| Transparency | Central Issue | Result of Transparency |
|---|---|---|
| Location | The location of an object in a distributed environment | User unaware of the location of services |
| Access | The method of access to objects in a distributed system | All objects are accessed in the same way |
| Migration | The re-location of an object in a distributed environment | objects may move without the user being aware |
| Concurrency | Shared access to objects in a distributed environment | Users do not have to deal with problems of concurrent access |
| Replication | Maintaining copies of an object in a distributed environment | Systems deals with the consistency of copies of data |
| Failure | Partial failure in a distributed environment | Problems of failure are masked from the user |

The important requirement to control CSCW applications in distributed systems is that there should be a clean separation between the mechanisms and the policies. The mechanisms are required for distribution management and suitable for the demands of CSCW. The policies govern the use of these mechanisms and meet

the control requirements. For example, there is a clear distinction between the ability to move an object (the mechanism) and the decisions about when the object should be moved and to which site (the policy) [15].

Hence, both the mechanisms and policies of distribution management should be suited more closely to the demands of group working. This raises some fundamental and, as yet, unresolved questions [15]:

- what are the most suitable mechanisms to support group working
- what are the appropriate management policies for a co-operative environment?

A solution to the problem will require a detailed understanding of both the behavior of distributed systems and the behavior of interacting user groups.

### 2.3.2. Evaluation of CSCW in distributed system

The needs of cooperative systems and the facilities offered by distributed systems present a number of problems. This sub section highlights a number of problems with existing distributed systems technologies. In particular, the following specific problems can be identified [18]:-

- **poor support for real-time group interaction**

A good support for highly interactive group activities is not provided by the distributed systems. The notion of a group simultaneously accessing shared data is generally not well catered for. In addition, systems do not provide the necessary real time responses for this class of application.

- **lack of support for multimedia cooperation**

Handling multimedia information in the distributed system is still in the beginning stages. This is particularly the case in group working where several users require access to multimedia information sources.

- **problems of distribution transparency**

Providing full distribution transparency may be misguided for CSCW applications. The problem with transparency is that decisions are made by the system on behalf of the application. The particular demands of CSCW require a greater level of control over the policies of managing the distributed system.

### 2.3.3. Similar Collaborative Systems

Collaborative tools can be Web-based or Software; divide by area-served divided into three categories [21]:

1. **Collaboration communication tools**

As unstructured interchange of information; send messages, files, data or documents between people and hence facilitate the sharing of information. Examples include: e-mail, Instant Messaging, faxing, voice mail and Wikis.

2. **Conferencing tools**

Refers to interactive work toward a shared goal; facilitate the sharing of information but in a more interactive way. Examples include: Internet forums, online chat, Telephony and Application sharing.

3. **Collaborative management tools**

Refers to complex interdependent work toward a shared goal; facilitate and manage group activities. Examples include: electronic calendars, project management systems, knowledge management systems, hosted intranet and online spreadsheets.

The collaborative tools for academic field needs a combination of the three categories to be powerful enough.

**Examples of Collaborative Tools:**

1. **eRoom**

Type: Collaboration Software. Platforms: Windows

Price: Commercial URL: http://www.eroomhosting.com/

2. **Teamspace**

Type: Collaboration Software. Platforms: UNIX, Windows

Price: Commercial, priced according to storage, number of team members. URL: teamspace.com/

3. **Basecamp**

Type: Web-based Collaboration Platforms: Mac, Windows, web browser: IE 6/7, Firefox, or Safari

Price: Commercial, priced according to chosen features URL: basecamphq.com

4. **CollaborativeWorkspaces**

Type: Web-based Collaboration. Platforms: all Web browsers, PDA's and smart phones

Price: Commercial. URL: collaborativeworkspaces.com

5. **iKE**

Type: Web-based Collaboration. Platforms: : Windows 2000/XP Pro, Internet Explorer 6.x, Firefox 1.x, Opera 8.x or higher

Price: Commercial, Priced according To storage and bandwidth URL: ike.com

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

usage.

At below a comparison between the pervious collaborative tools according to the tool supported features:

**Table 2 .2 comparison between the pervious collaborative tools according to the tool supported features**

| | discussion rooms | Files sharing | Task management | Scheduling | Shared board | Voting | Log File |
|---|---|---|---|---|---|---|---|
| eRoom | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| Teamspace | ✓ | ✓ | ✓ | ✓ | | | |
| Basecamp | ✓ | ✓ | ✓ | ✓ | | | |
| Workspaces | | ✓ | ✓ | ✓ | | ✓ | |
| iKE | ✓ | ✓ | ✓ | ✓ | | | |

## 2.4. Conclusion

In conclusion, this chapter has provided some background to the area of CSCW .It covers basic CSCW classes and characteristics. An overview of the CSCW general classes are presented including message systems, computer conferencing, meeting rooms, co-authoring , argumentation systems and task management . In addition, characteristics of CSCW including the form of cooperation and the geographical nature are demonstrated. Association between CSCW and Distributed Systems discussed as important issue. In addition to demonstration of requirement and style of control for cooperation in CSCW systems, it discussed specific problems in the

facilities offered by distributed systems. Besides, the mechanisms and policies of distribution management have presented. Based on this study, we will design a new web based tool serving graduation projects. The next chapter will discuss the analyses of this tool.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

# 3. System Analysis

The Cooperative frame work for Graduation Project (CFGP) web based tool provides a frame work to help team members to manage their various shared tasks without effects of to be at a different places. It equipped with many features such as discussion spaces, graphical board sharing and file sharing in a way to achieve more security, more flexibility and effectiveness communication.

This chapter gives the analysis of our requirements with respect of our environment to meet our needs in the design of CFGP web based tool.

## 3.1. Intended Environment

CFGP web based tool allows a group of members to work with each other electronically with high effectiveness of online interaction for serving the following parties:

- **University and Institute:**

University and institute may propose their students to use CFGP web based tool in their graduation project. The CFGP web based tool allow powerful communication with a flexibility in working time . For instance, CFGP web based tool if applied on graduation project, it assures that students are able to apply, analyze, synthesize and evaluate information in an online interaction. CFGP web based tool will serve in general academic field either in Baccalaureate, Master or Doctorate projects.

- **Research Department:**

Research Department can use CFGP web based tool for performing scientific research on questions posed by scientific theories and hypotheses. By using CFGP web based tool, the members can present their observations, criticisms and opinions more easily, also write and edit research paper and scientific reports. Moreover, the member can conclude their meeting as in real environment.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

## 3.2. Graduation Project Framework Characterization

The Graduation Project is designed to allow each student with the opportunity to go beyond the traditional educational program. By transferring learned skills to self-selected areas, students will expand their personal knowledge; apply learning to practical, real-life situations; and become lifelong learners. Students may choose their own topic from a variety of project proposals, or suggest another one and get it approved. To make sure that a high expectations are established for all students, teachers must clarify their expectations and agree on uniform standards.

CFGP web based tools analysis based on questionnaire that designed to assess the current used programs and recognize the problems and requirements for home working of graduation projects. The questionnaire completed by the targeted sample of graduation project students. Analysis of survey findings show that the graduation project work is usually doing as cooperative home working .Thus, the home working need specialized tools for coordinating and controlling the activities between team members and their supervisor.

The results of the survey indicate the following problems:

- **Connection problems:**

    1. Slow communication.
    2. The disconnection specially through sending files.
    3. Limited size for sending files.
    4. Dependency of being online for receiving files

- **Controlling and coordination problems:**

1. Communication programs(e.g. messenger ) designed for general communication and may confuse the member, which need to be in work environment in terms of being connected only with team members and availability of specific tools.

2. Difficulty of exchange fast notes over send new e-mail, or begin conversation.

3. Difficulty of maintains the up-dates of shared work.

- **Others:**

1. Difficulty of control and coordinate team work, distinguish of tasks and follow up the work progress. For example, the coordinator must be significantly has defined control tasks that will be more suitable to been supported by used tool.

2. The cost of tools.

Moreover, Survey Findings show the following requirements of specialized tool for graduation project home working:

1. Shared board for post notes, ideas, outlines, etc that must be accessed and seen by all.

2. Show edited- Planning time schedule.

3. Notify members for update.

4. Shared folders for members and private for supervisor.

5. Being useful for manage the project with supervisor and team members.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

According to our study, the general tasks of supervisor, coordinator and member are shown in the following sections:

- **Supervisor**

Several tasks are assigned to the supervisor during the project, which are:

1. Being available to have a discussion with a graduation project team member, if needed.
2. Providing a guideline to teams members, reviewing the report and giving grade to students.

- **Coordinator**

Team members may indicate one member as coordinator. This coordinator had been assigned to a number of responsibilities which are:

1. Planning time schedule that contains the tasks distributed on the weeks; to meet deadlines and manage the project successfully.
2. Periodically checking with their supervisor and demonstrates the finished project phases before submission.

- **Member**

Other individuals supposed to perform the following tasks:

1. Working on his assigned task.
2. Participating with other members by discussion and co-work on applied work.
3. Following the timeline for completion of all components of the graduation project.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

## 3.3. Non-Functional Requirements

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors.

In this part we could declare the non-functional requirements of the system as following:

- **System Constraints**: CFGP web based tool can display with any browser in any OS platforms. However, the system must supported by Internet / WAN connection. CFGP web based tool designed using Java Jdk 1.5.0 or later, Apache Tomcat 4.1 or later and Java Virtual Machine(JVM) .

- **Submission time**: Submission the report and the CFGP web based tool will be at the end of 1429H second semester inshallah.

- **Size of CFGP ( choose other word )**: the size required by CFGP web based tool is static at the client side. While on the server side, the size will be increase in term of increasing the size of folders being shared and data base.

- **System Constraints**: Portability or Mobility, Since CFGP is a web site, so it is moveable which allows any student to keep with the project working whether at home, office, at school and while travelling.

- **User requirements:**

    1. Tool should be as user-friendly as possible.

    2. Tool should be straightforward to use.

    3. A user manual should be provided for the user.

## 3.4. Functional Requirements

The system must serve the supervisor, team members and coordinator to meet their goals. Thus, an efficient cooperative framework for graduation project (CFGP) web based tool designed to provide an effective support for the dynamic control of the co-operative work in graduation project. This part will illustrate the features and actions provided in CFGP framework to serve the supervisor, team members and coordinator group.

Many features supported by CFGP web based tool to perform well-organized work:

- **Discussion rooms:** This discussion room is text chatting that can be classified into two rooms:

1. Private discussion room: for team members and coordinator discussion.

2. Open discussion room: for all, team members, coordinator and supervisor discussion.

- **Shared folders**

1. Team work folder: contains working files, such as codes, resources, versions of reports.

2. Supervisor folder: contains two subfolders :( From) contains files retrieved from supervisor and (To) contains files transferred from the team members to the supervisor.

- **Advertising space:** Enable the team member and supervisor to post their important notes to all members such the deadline.

- **Contacts list:** Shows the members who are online.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

- **Log file:** Records last events (e.g. Adding vote, Adding task) with time and members name

- **Task management (scheduling):** Enables the coordinator to schedule tasks.

- **Vote module** Enables members to vote by ”yes or no ” for the meeting that suggested by coordinator or supervisor . Then, the system display the result before one day of the meeting suggested day according to their choice.

- **Shared board:** Enables team members and supervisor to explain their notes in graphical board.

To start use CFGP, all users respect to do the following:

1. Connect to the network.

2. Open the tool and entering the user name, password and group name.

To Specify the actions provided by the tool for each user:

## ❖ At the members' side

1. Seeing which of group members and supervisor are online from the contact list.

2. Accessing and managing the Team work folder which contains teamwork such as the report and program codes.

3. Viewing tasks schedule.

4. Adding announcement to the advertising space.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

5. Viewing log file which contains the last ten modifications.
6. Opening received files from the supervisor that stored in From Supervisor folder.
7. Accessing the discussion room which contains text chatting.
8. Accessing the questionnaire (vote module).
9. Accessing the shared board.

### ❖ At the coordinator's side

The coordinator allowed by the system to perform the same actions as a members. Moreover, it had been allowed to be the administrator and do the following:

1. Creating the workspace.
2. Managing Supervisor folder e.g. Send works and files to the supervisor.
3. Managing teamwork such as dividing the work, determining the submission time and adding them to tasks schedule or edit task and delete it.
4. Create questionnaire (Vote module) for determining the supervisor meeting.
5. Delete the workspace.

### ❖ At the supervisor's side

1. Noticing who are online from the contact list.
2. Adding announcement to the advertising space.
3. Create questionnaire (Vote module) for determining the supervisor meeting.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

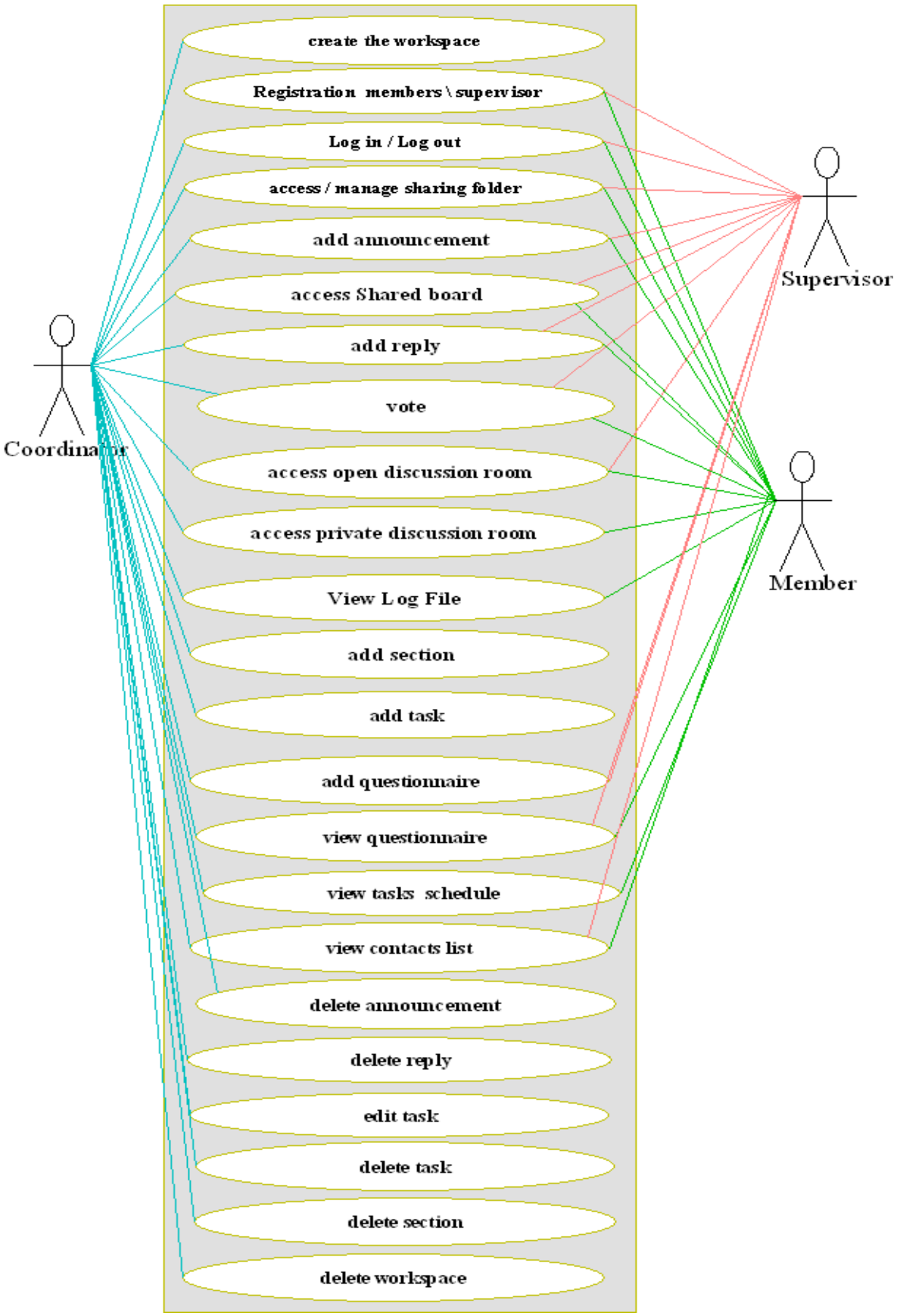
4. Accessing an open discussion room with any member.
5. Viewing received files from the group.
6. Modifying received files from the group.
7. Sending files to the group.
8. Accessing the shared board.

## 3.5. System Specification

The system specification specifies and establishes the methods of the CFGP web based tool to be used to ensure that each requirement has been met. This part presents a complete description of the functions to be performed by CFGP web based tool.

### 3.5.1. Use Case

The use case shows a collection of possible scenarios related to particular goals. Thus, the use cases for CFGP established to describe the interaction between a primary system actors and the system itself, represented as a sequence of simple steps. As discussion earlier, the CFGP system actors are: coordinator, members and a supervisor. Figures 3.1 show the use case activities significance to each user.




قسم مهارات تطوير الذات

عمادة السنة التحضيرية

**Figure 3. 1 : CFGP Use Case**

## 3.5.2. Expanded use case

The expanded use case which is a more detailed description of the events used to accomplish the system function. For deepness understanding of the system behavior, the expanded use cases constructed.

❖ **Expanded use case: Create The Workspace**

**Actor**: coordinator.

**Overview**: After established connection, the actor fills the registration form to create his /her account and his/her group work space.

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins after opening CFGP web based tool and click on "add new group to site" link. | |
| | The system establishes a connection to the server. |
| Fill the registration form then click "Register". | |
| | The system accepts, and creates the chosen name Workspace and gives the actor all authorities. |

❖ **Expanded use case: Registration Members \ Supervisor**

**Actor**: member and supervisor.

**Overview**: The actor fills the registration form to interact with her/ his group.

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins when the actor click on "be a new member of a group" link. | |
| | The system asks for entering the group name and password. . |
| The actor insert group name and password then click "Join". | |
| | After system authenticate of group and password, Join to group. Then , open new form contains information about actor. |
| Fill the registration form then click "Register". | |
| | Accept and give the actor all authorities. |

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

❖ **Expanded use case: Login/Logout**

**Actor**: Members, coordinator and supervisor.

**Overview**: After established connection, the actor enters his user name, password and group name to log in. finally, the actor will choose log out to quit from tool when finishing his work.

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins when the actor opening CFGP web based tool home page. | |
| | The system establishes a connection to the server and asks for entering the actor name, password and group name. |
| The actor insert his name and password and group name then click ”Log in” icon. | |
| | After system authenticate of user name, password, and group name open the group page and begin responding to the user actions. |

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

| | |
|---|---|
| Request logout by click on ”logout” icon | |
| | Quit the CFGP web based tool and close connection to the server. |

- **Expanded use case: View Contact List**

**Actor**: Members, coordinator and supervisor.

**Overview**: show list of online team members.

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins when the actor connects to his/her group page in the CFGP web based tool. | |
| | The system displays the contact list on the screen automatically. |

- **Expanded use case: Accessing / managing shared folder**

**Actor**: Members, coordinator and supervisor.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

**Overview**: Open the shared folder which contains Team Work folder and Supervisor folder (which contains To / From folders). Then, the actor can download, rename, delete, create, move, copy, upload, view and edit files.

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins when the actor wants to view shared folder and clicks on shared folder icon | |
| | The system opens the folder on the server. Then, the folder contents will appear on the screen. |
| The actor can download, rename, delete, create, move, copy, upload, view and edit files. | |
| Click close icon to close folder. | |
| | Save the changes on the server, and then close the folder. |

- **Expanded use case: Add a section in advertising space**

**Actor**: coordinator

**Overview**: Actor can add new section of announcements

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants add a new section. | |
| | Open window contains text area for allowing the actor to type the title and section information. |
| Type the contents of the section and click ”add section” button. | |

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

| | |
|---|---|
| | Accept and save to server |

- **Expanded use case: Add announcement in advertising space**

**Actor**: member, coordinator, supervisor

**Overview**: the actor can add announcement to strike the sight of other actors.

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants add a new announcement. | |
| | Open a new window contains text area for allowing the actor to type the new announcement information. |
| Type the contents of the announcement and click OK. | |
| | Accept the announcement and view it in the announcement space. Proceeding by actor name. |

- **Expanded use case: Add replay in advertising space**

**Actor**: coordinator, member, supervisor.

**Overview**: the actor can add reply for announcement and discussion with anthers actor about this announcement.

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants add reply. | |
| | Open new window contains text area for |

| | |
|---|---|
| | allowing the actor to type the reply |
| Type the contents of the reply and click ”submit” button. | |
| | Accept and save to server. |

❖ **Expanded use case: Deleting section from advertising space**

**Actor**: coordinator

**Overview**: the actor can delete exist section.

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants delete a section. | |
| | The use case begins when the actor wants delete a section. |
| The actor click on ”delete” link. | |
| | Delete section from server. |

❖ **Expanded use case: Deleting announcement**

**Actor**: coordinator

**Overview**: Actor can delete exist announcements

**Typical course of event:**

| Actor action | System action |
|---|---|

| | |
|---|---|
| The use case begins when the actor wants to delete an announcement. | |
| | Open a new window contains announcements and allowing the actor to choose ”delete announcement” link. |
| The actor click on ”delete” link. | |
| | Delete announcement from server. |

- **Expanded use case: Deleting reply**

**Actor**: coordinator

**Overview**: Actor can delete exist reply

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants to delete a reply. | |
| | Open a new window contains announcements and allowing the actor to choose ”delete rply” link. |
| The actor click on ”delete” link. | |
| | Delete announcement reply from server. |

- **Expanded use case: View log file**

**Actor**: coordinator, member ,

**Overview**: open log file which contains the last ten actions in format of (actor name, action, date and time).

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins when the actor want to view the log file and click ”log file” icon. | |
| | The system opens a window and the content of log file will be appear on the screen |

- **Expanded use case: Access open discussion room**

**Actor**: member, coordinator, supervisor.

**Overview**: actors can make a discussion with their supervisor in this room.

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants to enter a discussion room and click ”Discussion room” icon. | |
| | The system opens a window of a discussion room. |
| Write text notes then click send button. | |
| | Receive text notes from actor, then send the notes to all actors which they existing at the discussion room. |

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

❖ **Expanded use case: Access private discussion room**

**Actor**: member, coordinator.

**Overview**: actors can make a discussion with each other in this room.

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants to enter a discussion room and click ”private discussion room” icon. | |
| | The system opens a window of a discussion room. |
| Write text notes then click send button. | |
| | Receive text notes from actor, then send the notes to all actors which they existing at the discussion room. |

❖ **Expanded use case: Add Task**

**Actor**: coordinator

**Overview**: the actor can add tasks to schedule to show important activities (tasks) for project on one screen.

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants to Add a new task and click on ”Add task” icon. | |
| | Open a new window contains text area for allowing the actor to type the new task and its submission time. |
| Type the contents of the task and | |

| | |
|---|---|
| determine the submission time. Then, click OK. | |
| | Accept the tasks and view it in the tasks schedule. |

- **Expanded use case: Edit Task**

**Actor**: coordinator

**Overview**: Actor can edit a task

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants to edit task and click on ”edit” link. | |
| | Open a new window contains text area for task that want edit it , allowing the actor to edit the task and its submission time |
| Edit the contents of the task and determine the submission time. Then, click OK. | |
| | Accept the tasks and view it in the tasks schedule. |

- **Expanded use case: Delete Task**

**Actor**: coordinator

**Overview**: Actor can delete task

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants | |

| | |
|---|---|
| to delete task and click on ”delete” link. | |
| | Delete task from server. |

- **Expanded use case: Add questionnaire (vote module)**

**Actor**: coordinator, supervisor.

**Overview**: the actor can add questionnaire for meeting time with supervisor.

**Typical course of event:**

| Actor action | System action |
|---|---|
| The use case begins when the actor wants to add new questionnaire (vote module) and click ”vote creation” icon. | |
| | Open a new window contains date and time that suggested supervisor meeting. |
| Select the date and time which they suitable for meeting then click ”submit” button. | |
| | Add the questionnaire in its space for previous suggested date and time. |

- **Expanded use case: Vote of questionnaire**

**Actor**: member, coordinator, supervisor.

**Overview**: the actor can select choice from questionnaire for meeting time.

**Typical course of event:**

| Actor action | System action |
|---|---|

| | |
|---|---|
| The use case begins when choose one of the choices and click ”submit” button. | |
| | Accept the choice from actor and open new page which displays the number of members choose "yes" and number of member choose "No". |

- **Expanded use case: View questionnaire (vote module)**

**Actor**: member, coordinator, supervisor.

**Overview**: the actor can select view questionnaire

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins when the actor connects to his/her group page in the CFGP web based tool. | |
| | The system displays automatically the last new vote on the screen or displays the result of voting before one day of suggested meeting day. |

- **Expanded use case: View Task Schedule**

**Actor**: member, coordinator.

**Overview**: open tasks schedule which contains the tasks and their submission time.

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins when the actor wants to access Shared board and click ”Shared board” icon. | |
| | Open a new window contains Shared board and allowing the actor to use graphical board. |

- **Expanded use case: Delete Workspace**

**Actor**: coordinator

**Overview**: Actor can delete space

**Typical course of event:**

| Actor action | System action |
|---|---|
| This use case begins after opening CFGP web based tool and click on ”Delete group” button. | |
| | The system accepts, and deletes the chosen name Workspace from server. |

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

## 3.6. Conclusion

In conclusion, this chapter covered the analysis of CFGP web based tool requirements. Furthermore, the non-functional requirement which impose constraints on the design or implementation had been described. Moreover, the features and actions that offered by CFGP presented to serve the supervisor, team members and coordinator of the group. Finally, it includes a set of use cases that describe all of the interactions between the users will have with and the tool. Next chapter will introduce the appropriate design of CFGP web base tool.

# 4. Design

In order to distribute and serve the workspace of graduation project as a CSCW application, the CFGP tool needs a database management system, with a powerful support, in addition to what serves the graduation project management .These needs and techniques will be chosen and discussed within this chapter. The main contents of this chapter will include technical design of CFGP components, network architecture, network protocols and database analysis.

## 4.1. CFGP General Architecture

CFGP is a CSCW web based tool enable team member of graduation project to achieve the cooperative work in an appropriate environment. The figure (4.1) shows

the CFGP architecture which divided into two primary components: client side and server side.

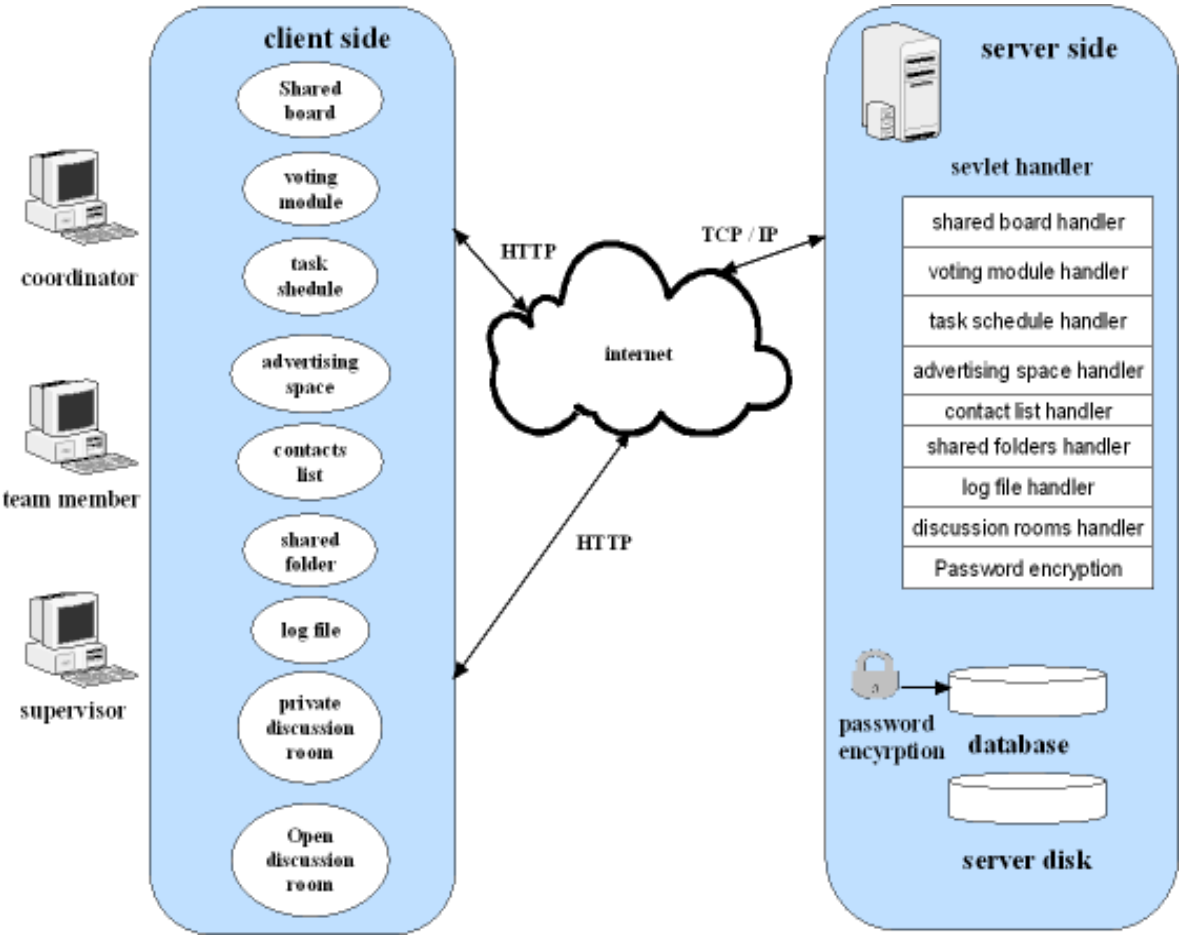



**Figure 4. 1: CFGP Architecture**

## 4.1.1. Client Side Architecture

Through web browser, graduation project team members view their workspace environment. Following sections give description details about each CFGP user workspace component.

### 4.1.1.1. Text Discussion Rooms

It is a virtual platform to facilitate and manage real-time text messages .Once a group registration complete, the two group's rooms created and its name registered in

chat property file as group name. When a group member clicks on discussion room icon, chat page will displayed by getting the room name that is defined by the session. The internal chatting process depends on some characteristics of web pages, such as, automatic refreshing of web page used to check for instance messages submission or logged in of other members. Instance messages processed as HTTP request over TCP/IP protocol of Internet. Web server handles these requests and generates a response of discussion rooms as dynamic JSP pages.

#### 4.1.1.2. Files Sharing

CFGP tool server holds the 'Shared Folders' directory which contains subdirectories, one for each group, created by each group registration and titles by group name. Each group can upload, download file or group of files and directories by compress them over server, through HTTP protocol and by using functionality of HTTP headers for application layer, and TCP/IP for transport layer. Also, the members can edit, copy, delete and transfer file to other directory.

#### 4.1.1.3. Shared Board

Text and graphical shared board classified into public and private as team's needs. CFGP shared board design to be executed at client side where HTTP represents the protocol of application layer .Once a team member draw through some interactive based web application that use the member's web browser to provide a user interface. The drawing attributes submitted to sockets .Then server get drawing attributes and send them to all joined team members where their drawing will be done on their browser depending on these attributes. Communication represented through TCP/IP protocol at transport layer from TCP Sockets of clients and server.

#### 4.1.1.4. Log File

A log file is file that lists last actions of collaboration work. Log file processing depend on record each considerable action in log file table of CFGP database. Log file table contains action description, actor, time and date. Server accesses log file table of

database in order to generate log file as web page. Log File keep only the last ten records of actions for each group, all group's records except the last ten records will deleted from the table to avoid maximize database size without real need.

### 4.1.1.5. Advertising Space

Advertising space is interactive web pages where a member can review advertisement left by others and leave a replay. These web pages are dynamically created and access database within server side. See figure (4.2)

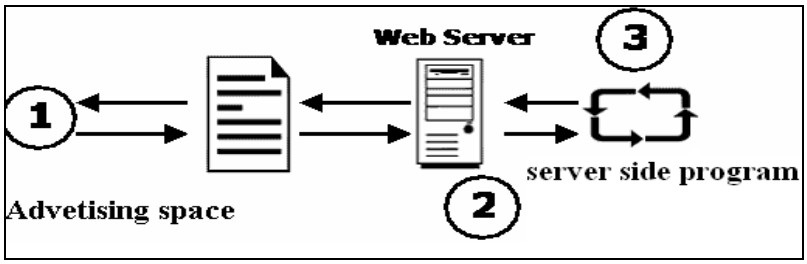



**Figure 4. 2: Advertising space.**

The following statements explain the steps of advertising space design as shown in figure 4.2:

1. Team member sends request. This will be submission of new advertisement or adding comment of old one within web page.

2. The web server receives this request and determines through its configuration files that the web page must be first processed by another program adding dynamic HTML content to a web server application.

3. This program executes the commands in the web page and returns an HTML document to the web server that returns the HTML page to the user. Processing of CFGP advertising space deals with a table of CFGP database. Advertisement table contains the name of advertiser, advertisement title, advertisement content, time and date.

### 4.1.1.6. Voting Module

Voting for a project team added by a supervisor or coordinator to make a decision about appointment of meeting, it is closed before 24 of selected appointment. This mechanism operated by accessing a database located at server. A vote table contains the name of who create that vote, time and date of meeting and count of vote answer (yes or no).Once one day before the suggested appointment of meeting the result automatically displayed when a user connects to his/her group page in the CFGP web based tool. The system will display if there is a meeting or not according to the percentage of AYES (members vote yes) and the percentage of NOES (members vote no)

### 4.1.1.7. Task Management

As a dynamic web page, tasks schedule created by HTTP response of HTTP request posted by coordinator or supervisor. HTTP request contains tasks information like content, deadline. etc .Web server insert these information to database. Then server access task table of CFGP database in order to create the task schedule as dynamic web page for each group. Task table contains task, time and date.

### 4.1.1.8. Contacts List

Once a members log in the CFGP site, their names inserted into table of CFGP database, and dropped when they log out. If web page refreshed contacts list dynamically check that table and get current instance of contacts.

## 4.1.2. Server side architecture: Data base design

Server of CFGP tool manages many dynamic web pages and data which stored in shared database. Database design issues are shown in following sections:

### 4.1.2.1. Data base requirements

Many groups of members are included in CFGP tool. Each group contains many users. Users divided to three types (Supervisor, Coordinator or Members). According to the user type, every user can do more than one function. So, CFGP tool needs a huge of stored data to achieve these functions. In addition data is changing, or new data is being generated continually. To solve this problem, CFGP tool uses a database storing records of data. Each record consists of several fields. Some of the fields are special"key" fields, which provide fast searches on database. CFGP database includes:

**Table 4. 1CFGP Database Tables**

| ***Table*** | ***Description*** |
|---|---|
| Project | Records information about the **Project**: (Group Name and password) |
| User | Records information about the **User:**<br>(user name, password, E-mail, type, avatar, title and signature)<br>Users are divided into three types:<br>(Supervisor, Coordinator and Member). |
| Log File | Records information about the **Log File:**<br>(log number, date, time and log content) |
| Task | Records information about the **Task:**<br>(task number, time, date, and task text) |
| Forums | Records information about the **Forum:** ( forum_id , info and title) |
| Thread | Records information about the **Thread:** (thread_id , views and title) |
| Message | Records information about the **Message:**<br>(id, replay_id, date and message) |
| Vote | Records information about the **Vote:** |

| | |
|---|---|
| | (vote number, creation vote name, time, date , isNo , isYes) |
| List | It Records information about the **List:** (name and type) |

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

### 4.1.2.2. ER model

The conceptual model represented by drawing ER model (Entity-relationship model) having eleven entities (Project, User, Log File, Task, Forum, Thread, Message, List and Vote) with different attributes. Supervisor, coordinator and member are disjoint subclasses of User. See figure 4.3.

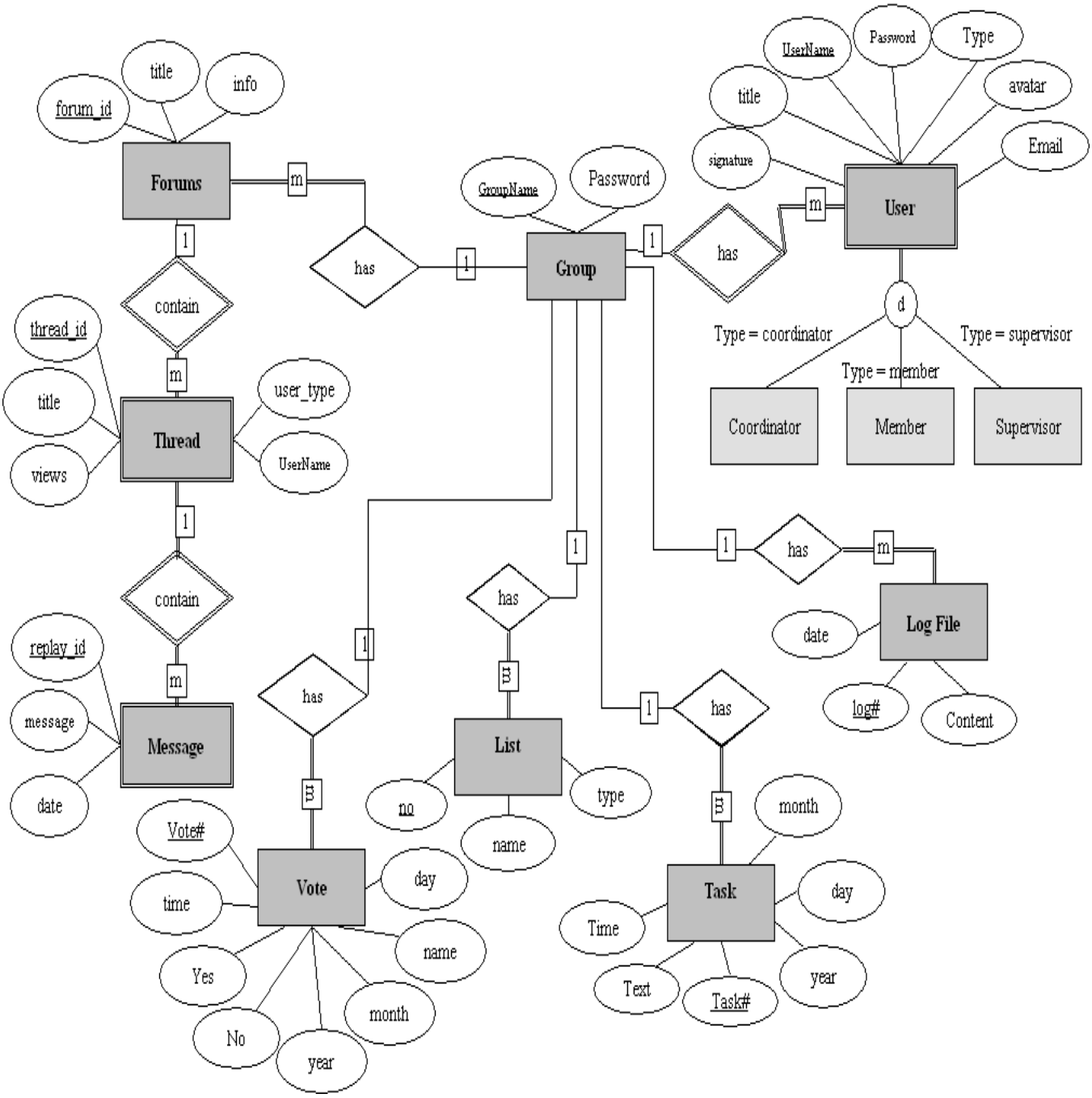


**Figure 4. 3: Database ER Model.**

### 4.1.2.3. Mapping

The conceptual data base design (ER model) mapped into the logical model below. See figure 4.4.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

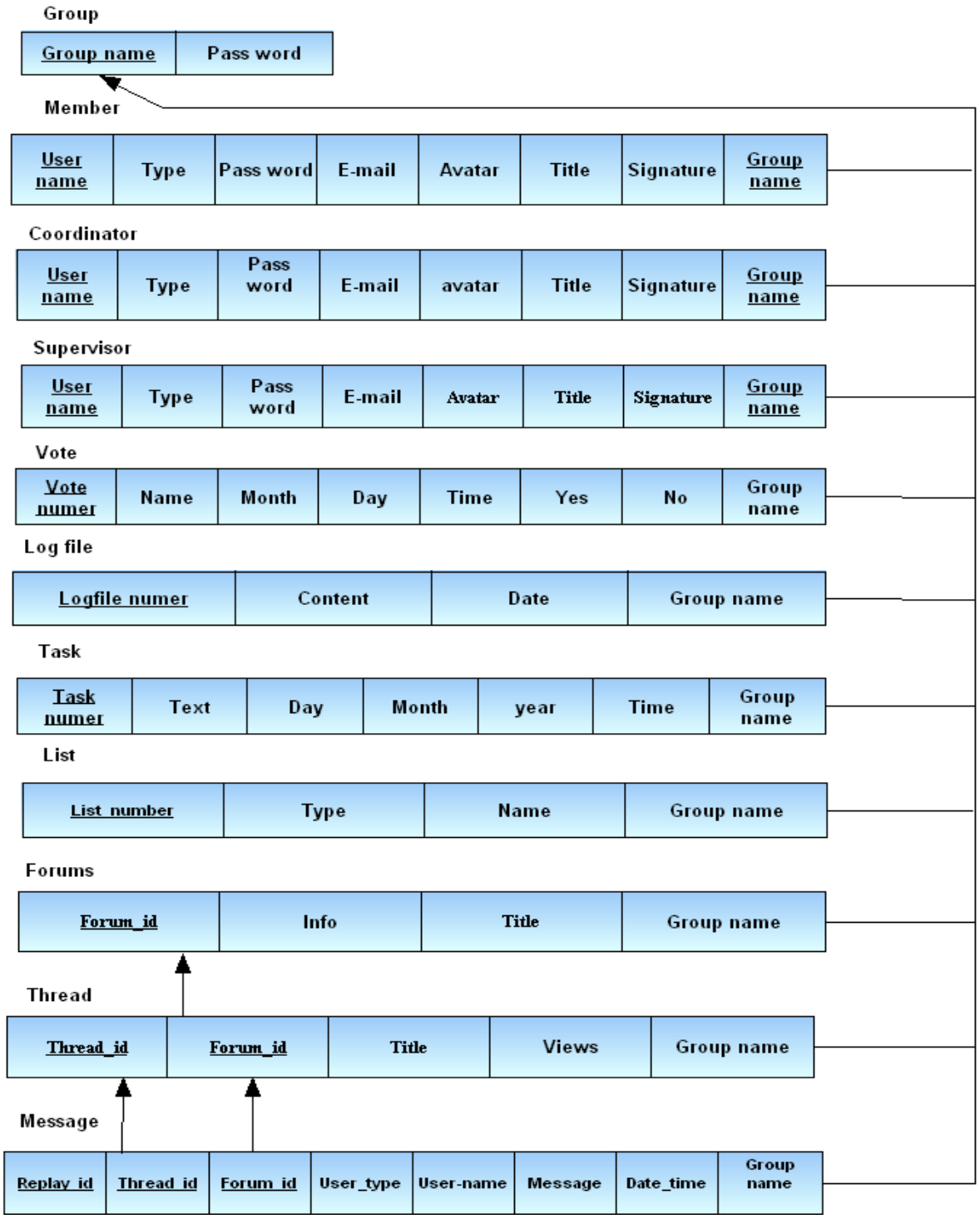



**Figure 4. 4: Database Mapping.**

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

### 4.1.2.4. Security and Cryptography

Cryptography is an abstract or concrete operation that performs a security-related function and applies cryptographic methods. Cryptography is widely used for secure application-level data transport. Cryptography traditional goals are: data confidentiality, integrity, and authentication. In shared key encryption, two parties know the same cryptographic key. The sender (System A in Figure 4.5 Shared Key Encryption) encrypts the data with the key to create encrypted data. The recipient (System B in Figure 4.5 Shared Key Encryption) decrypts the encrypted data with the same key. Since only a holder of the cryptographic key can decrypt the data, the encrypted data can be transmitted across the network without being understood by other parties [22]. See the following figure.

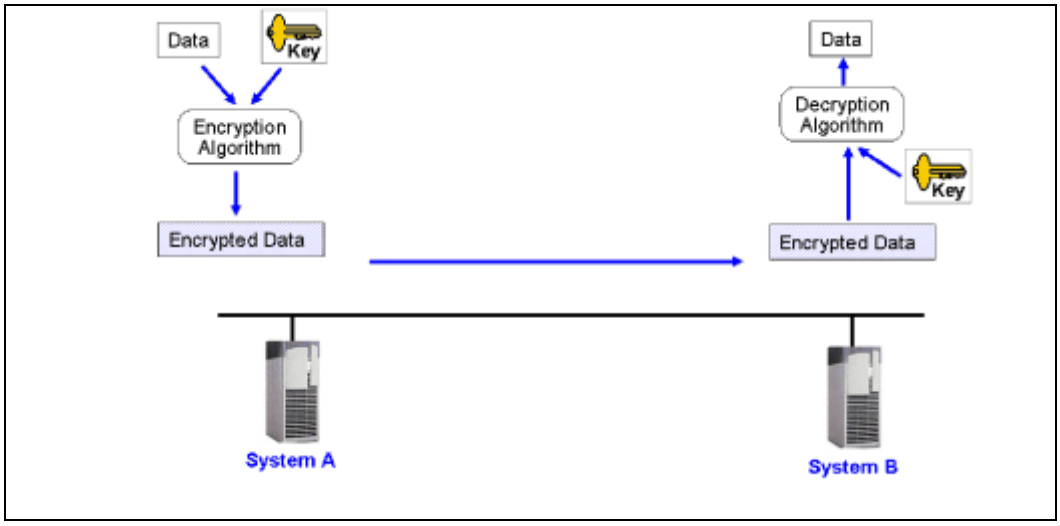



**Figure 4. 5:Shared key encryption.**

CFGP tool applies shared key encryption algorithm on team members' passwords where shared key encryption algorithm processing speed is so fast. Once a team member registers, its password encrypted and stored in CFGP database as encrypted password. However, when team member log in ,its submitted password get to server as encrypted one then it will be compared with stored encrypted password in CFGP database to be authenticated .

## 4.2. Communication model

The common examples of area network types are: A wide area network (WAN) and a local area network (LAN). CFGP tool designed to be used over WAN since the nature of group work together and scattered at regions or countries. The networking components of CFGP are based on Client-Server model. According to these architectures, the system will use appropriate protocols. CSCW applications are bound to use several protocols to support common layers for collaborative tools. Some protocols provide services like shared files, shared board, discussion rooms, and integration for both synchronous and asynchronous applications.

### 4.2.1. Network Model (Client-Server Architecture)

At Client-Server Architecture, client sends requests to servers every time it is instantiated and the servers accept, process, and return requested information to the client over the same network that the request was sent. CFGP servers come in different types: web servers, and database servers. Clients are the web browsers which allow users to interact with text, images, and other media on a web page. (See figure 4.6).

Servers hold private resources and content which assigns to each group (e.g. shared files, posts adverting, voting results ... etc) or public contents of the website (e.g. databases of registration users and groups). Servers also control the access to the group resources by use the variety of authentication techniques. These techniques include login requests that verify the users' identity through the use of a registered username and password where they are stored in a database.

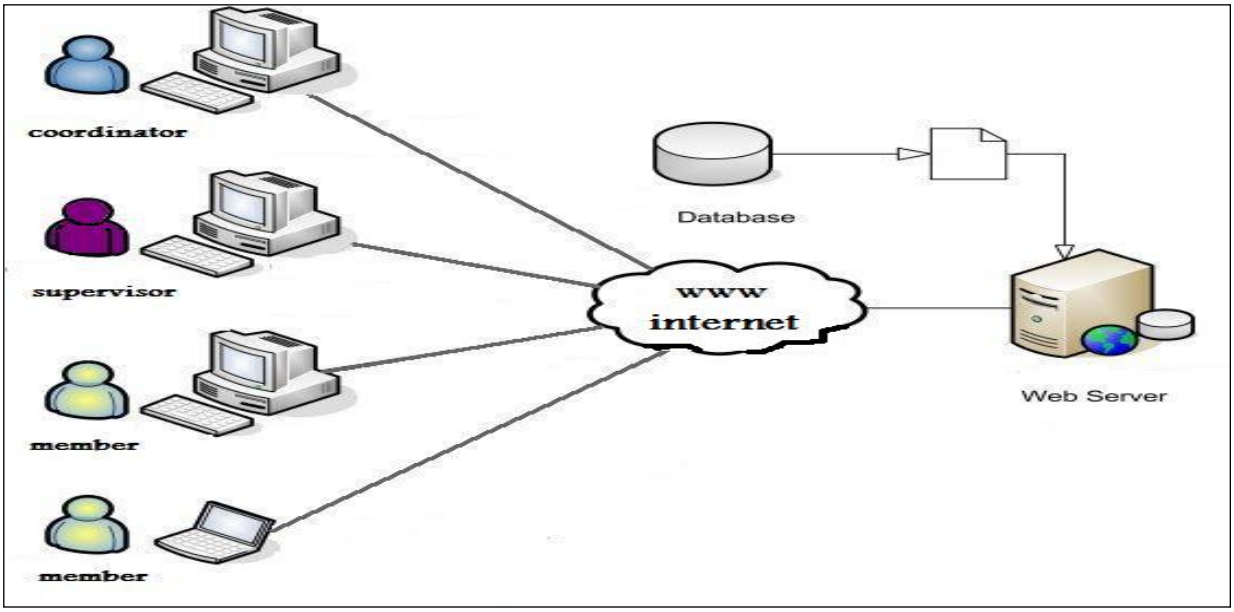



**Figure 4. 6:Client-Server Model.**

## 4.2.2 Network Protocol (HTTP, TCP/IP)

CFGP tool in general, build on HTTP protocol for application layer, and TCP/IP for transport layer. Hypertext Transfer Protocol (HTTP) is a client-server protocol by which two machines can communicate over a Transmission Control Protocol / internet protocol (TCP/IP) connection [23]. An HTTP server is a program that sits listening on a machine's port for HTTP requests. An HTTP client opens a TCP/IP connection to the server via a socket, transmits a request for a document, then waits for a reply from the server. Once the request-reply sequence is completed, the socket is closed. The lifetime of a connection corresponds to a single request-reply sequence. (See figure 4.7).

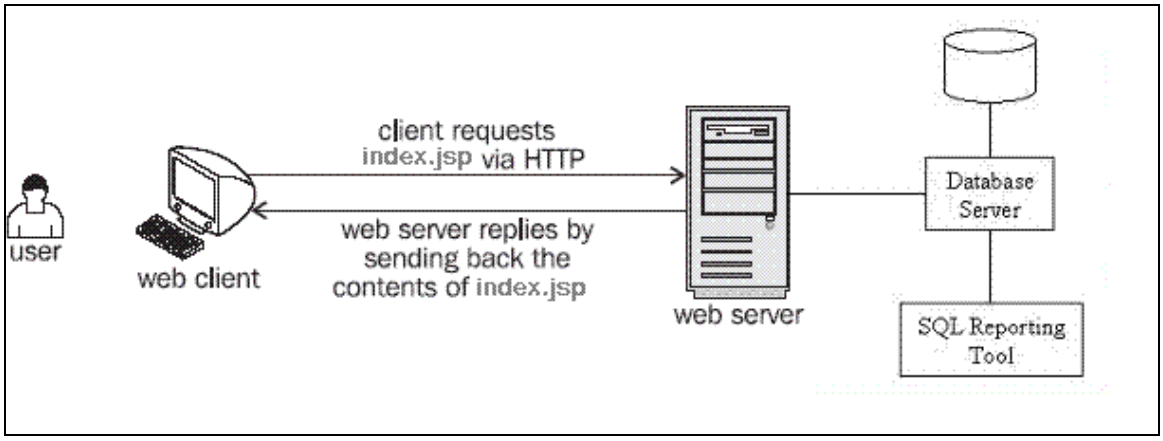



**Figure 4. 7:HTTP**

## 4.3. GUI design

A GUI used to enable user to interact with the contents of the site as required. Designing the GUI is an important part of application programming, aims to enhance the efficiency and ease of use of program, a design discipline known as usability. Techniques of UI design are used to ensure that the visual language introduced in the design is well tailored to the tasks it must perform. The CFGP's GUI aims to reach and achieve the guidelines of web usability to be absolutely sure that it is easy to use, work properly, and ultimately attract and keep users. CFGP GUI consists of two parts:

1. CFGP site: deal with registrations and logins queries (See figure 4.8 and 4.9).

2. Group's workspace: handle and cover the group works (See figure 4.10).

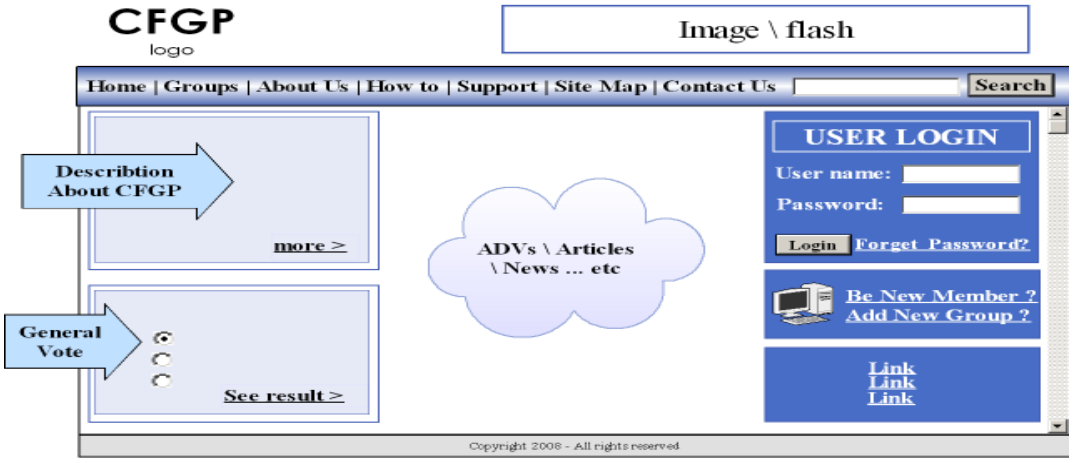



**Figure 4. 8:CFGP home page design.**

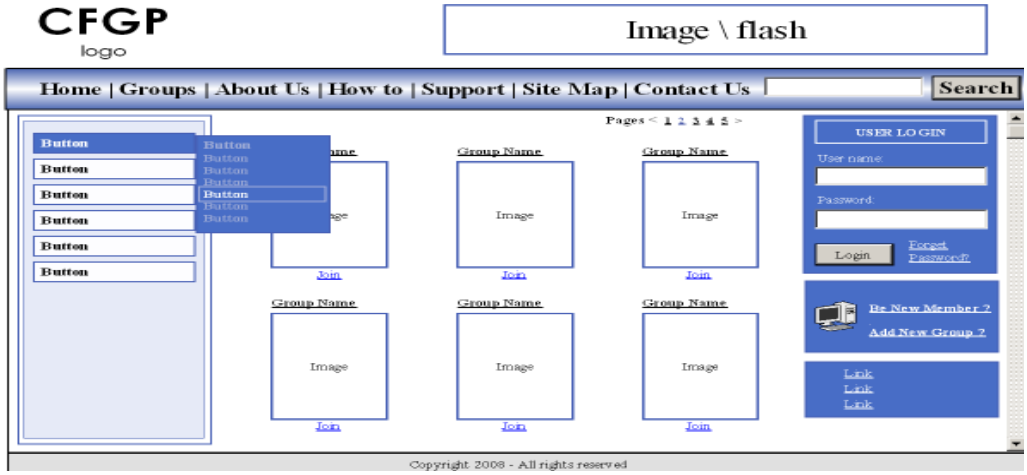



**Figure 4. 9: CFGP groups page design.**

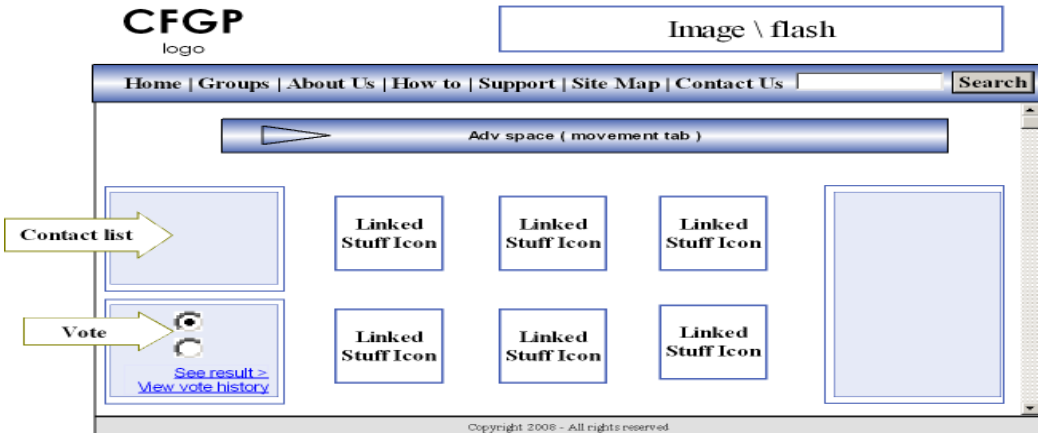



**Figure 4. 10: CFGP group workspace page design.**

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

## 4.4. Conclusion

In conclusion, this chapter has provided a general overview of CFGP architecture that is divided into client side and server side. Client side architecture represents a user view of workspace environment that consist of text discussing rooms, file sharing, editors, shared board, log file, advertising space, voting module and task management. In other hand server side architecture contains a database and remote disk that response each client request. In addition this chapter describes the communication model and the GUI of CFGP. Next chapter will introduce the implementation of CFGP component.

# 5. Implementation

This chapter aims to provide implementation details of CFGP web based tool. Our implementation is a java web-based application connected with a mySQL database. The chosen comes for many reasons shown below.

## 5.1. Choice of technologies and techniques

There are a range of techniques and technologies that can be used to develop CFGP web based tool. The chosen carried out with the aim of develop the CFGP web based tool at a manner of usable, independent platform, extensible, secure, and all other features that support a powerful collaborative framework.

### 5.1.1. Software or Web-based application?

We choose to program the tool as a web based application rather than a software tool for many reasons. The group members can supplement their cooperative work using

any computer having a web browser and connect to internet. Because it is a web site, the group members do not need to setup the tool, which is heavily loaded, have limitations, constrict system requirements and waste time.

### 5.1.2. Programming language

Another challenge faced us is the selection of programming language. As we decide to build CFGP web based tool as a web application , the major programming language that is powerful for web tools is Java which is embedded in JSP , servlet or Php pages. We select java which is embedded JSP and servlet because it is [24]:

- **Object-oriented language** based on open , public standards. It has a much richer collection of core functions than any other computer language which usable to make the notation of structure and reusability clearer
- Java is **multithreading** that performs several tasks simultaneously. Multithreaded give efficient. With servlets, the Java Virtual Machine stays up, and each request is handled by a lightweight Java thread, not a heavyweight operating system process. With servlets, however, there are *N* threads but only a single copy of the servlet class. Servlets also have more for optimizations such as caching previous computations, keeping database connections open, and the like.

- Java is **distributed**. Distributed computing involves several computers working together on a network. Java is designed to make distributed computing easy. Since networking capability is inherently integrated into Java, writing network programs is like sending and receiving data to and from a file , that's help the members to deal and manage the cooperative works.
- The created programs are **portable in networks and platform independent**. The source program is compiled into what Java calls byte code, which can be run anywhere in a network on a server or client. Also,

java program can run on a wide rang of computers and devices so the member unbinding to connect with the team from one place.

- Java is **extensible**. New feature can easily add as future enhancement. Java was designed to adapt to an evolving environment. New class can be loaded on the fly without recompilation. There is no need for developers to create, and for users to install, major new software versions. New features can be incorporated transparently as needed to CFGP web based tool.
- Java is **secure**. As an Internet programming language, Java is used in a networked and distributed environment. If user download a Java applet (a special kind of program) and run it on computer, it will not damage the system because Java implements several security mechanisms to protect system against harm. In CFGP, the collaborative work concerned specific team members, the important data may encrypt and protect from attack
- **Convenient.** Servlets have an extensive infrastructure for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions, and many other such utilities.
- **Powerful.** Java servlets can talk directly to the Web server. This simplifies operations that need to look up images and other data stored in standard places. Servlets can also share data among each other, making useful things like database connection pools easy to implement. They can also maintain information from request to request, simplifying things like session tracking and caching of previous computations.

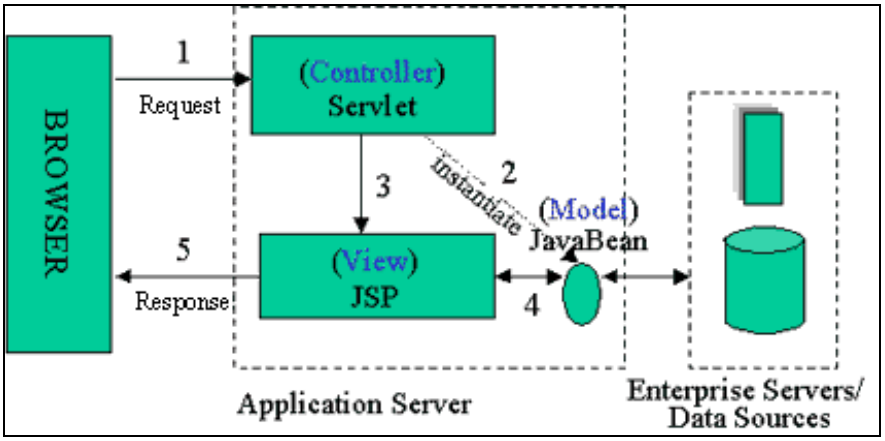



**Figure 5.1: use of servlet an JSP**

Figure 5.1 combines the use of both servlets and JSP. It takes advantage of the predominant strengths of both technologies, using JSP to generate the presentation layer and servlets to perform process-intensive tasks. In fact, this architecture will be suitable for applications which are more complex.

### 5.1.3. Data base System

MySQL is an open source, multithreaded, multi-user SQL database management system. MySQL is a cross-platform (support the most popular operating system: Windows, Mac OS X, Linux, BSD, UNIX and may support z/OS).

MySQL databases have many features that lead them to be suitable choice, the features are [25]:

- **Scalability and Flexibility :**Run deeply embedded applications with a footprint of just 1MB, or Massive data warehouses holding terabytes of information , unlimited database maximum size, and around 2GB maximum table size**.**
- **High Performance**: Ultra-fast load utilities, distinctive memory caches and full-text indexes.

- **High Availability:** Run high-speed master/slave replication configurations or specialized cluster servers offering instant failover.
- **Robust Transactional Support:** Complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability and multi-version transaction support.
- **Web and Data Warehouse Strengths:** High-performance query engine, tremendously fast data insert capability and Strong support for specialized web functions, like fast full text searches.
- **Strong Data Protection:** Powerful mechanisms for ensuring only authorized users have access, SSH [26] and SSL [27] support safe and secure connections, Powerful data encryption and decryption functions.
- **Comprehensive Application Development**: Support for stored procedures, triggers, functions, views, cursors, ANSI-standard SQL, and more. Also, Plug-in libraries to embed MySQL database support into nearly any application.
- **Management Ease**: Average time from software download to complete installation is less than 15 minutes.

MySQL requirement and supported tools to implement CFGP:

- **Drivers:** JDBC.
- **Application Development Language:** Java.
- **Connector Database Tools:** mysql-connector-java-5.1.6-bin.

## 5.2. System implementation:

The following section discusses the implementation of each of the individual components in CFGP web based tool. Our tool must be implementing as a set of cooperative operations in distribution manner.

### 5.2.1. System basic

In general there are many common languages and techniques applied in CFGP web based tool, as follows.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

- **JavaScript and HTML :**

JavaScript is a programming language that is mostly used in web pages, usually to add features that make the web page more interactive that allow putting dynamic text into HTML page and reacting to events. When JavaScript is included in an HTML file it relies upon the browser to interpret the JavaScript.

For example (**JavaScript):** To open page in new window we use window. Open (page name) function, and (**HTML):** To show image.

```
<td height="20" valign="top" >
<a href=" javascript:void(0)" onclick="window.open('memberView.jsp')">
<img src="image/Task.png" /></a>
```

- **Java applet**

Java applet is a program written in the Java programming language that can be included in JSP/HTML page. When using a Java technology-enabled browser to view a page that contains an applet, the applet's code is transferred to system and executed by the browser's Java Virtual Machine (JVM).( at the client side)

- **Java Packages:**

In CFGP web based tool we use some packages as follow:

**Table 5. 1 CFGP java used packages**

| *Package* | *Description* |
|---|---|
| **import** | the servlet classes that can extend the HttpServlet, and the http |

| | |
|---|---|
| **javax.servlet.http.*** | classes so, that allow to access the various parameters of the hypertext connection. We use some methods for example:<br>**public void** doGet or doPost<br>(HttpServletRequest request,<br>HttpServletResponse response ) **throws** ServletException, IOException{//….}<br>which takes two<br>parameters:<br>• An object containing information about the incoming request<br>• An object containing information about our response. |
| **import java.sql.*** : | Provides the API for accessing and processing data in a data source using the Java programming language like DatabaseMetaData and PreparedStatement . |
| **import java.sql.Connection** | A connection (session) with a specific database. We use some methods like getMetaData() and prepareStatement(String sql). |
| **import java.sql.ResultSet** : | A table of data representing a database result set, which is usually generated by executing a statement that queries the database. We use some methods like next(). |
| **import java.sql.Statement :** | The object used for executing a static SQL statement obtaining the results produced by it.We use some methods like executeQuery(Stringsql) and executeUpdate(Stringsql) . |

- **HTTP Session Tracking**:

Session tracking used because the interaction between clients and the server based on session management.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

Using sessions in java is straightforward and involves four basic steps [28]

**1. Accessing the session object associated with the current request**.

Call request.getSession to get an HttpSession object, which is a simple hash table for storing user-specific data.

To add data to the session regardless of whether data was there already, getSession() (or, equivalently, getSession(true)) is the appropriate method call because it creates a new session if no session already exists. So, use getSession(false), which returns null if no session already exists for the current client.

We apply this method in **log in**, see (log in).

**2. Looking up information associated with a session.**

These session objects have a built-in data structure (a hash table) in which can store any number of keys and associated values use session.getAttribute("key") to look up a previously stored value. The return type is object, so we must do a typecast to whatever more specific type of data was associated with that attribute name in the session .

We applay this method in (log file, advertising space, voting module, task management and contact list).

```
HttpSession session = request.getSession(true);
String name = (String) session.getAttribute("username");
String type = (String) session.getAttribute("type");
String group = (String) session.getAttribute("group");
```

**3. Storing information in a session**.

Use setAttribute method to associates a value with a name.

We apply this method in **log in**, see (log in).

**4. Discarding session data.**

Call removeAttribute to discard a specific value for entire session. Call logout to log the client out of the Web server and invalidate all sessions associated with that user.

We apply this method in **log out,** see (log out).

### 5.2.2. Registration:

In order to use CFGP web based tool, group and its team members must have an account on CFGP web site. Registration process classified into two situations as shown in figure 5.2

**Figure 5.2: Registration**

**1-create a new group account:**

- **At client side:**

Create a new group account assumed to be processed by group's coordinator. So, group's registration implies coordinator registration and vice versa see figure 5.3.

From browser coordinator fill the (Registration form) for required data. These data submitted as HTTP requests to registration servlet handler at server side to process them. See figure (5.3)

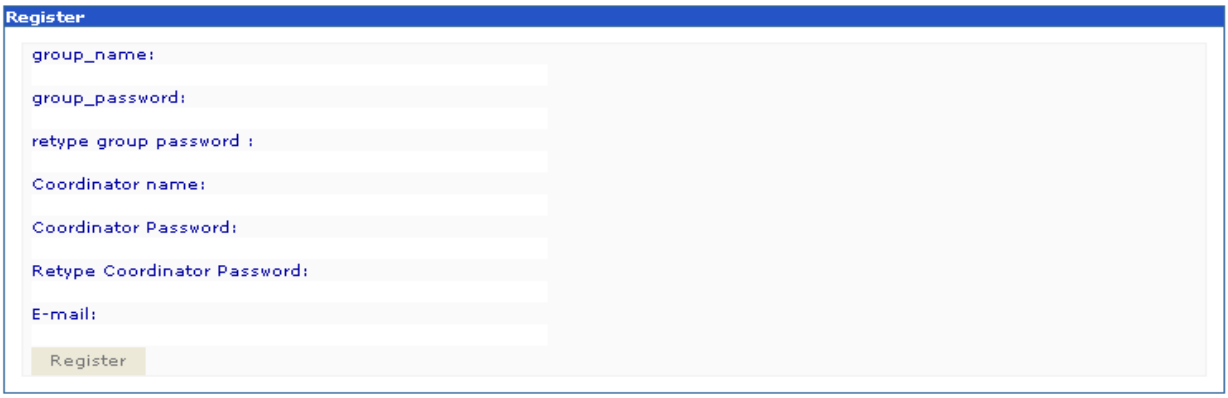



**Figure 5.3: new group registration.**

E-mail format is validated by java script function using regular expression concepts. As following:

```
function  isValidEmail(email){
 var RegExp =/ ^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$/
if(RegExp.test(email))
{  return true; }else{   return false;    }
```

❖ **At server side:**

For group and coordinator registration, program gets data as HTTP requests from a registration form of registration page. As following:

```
public void doPost (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
        PrintWriter out = response.getWriter();
    try{
   String RegUser = request.getParameter("user");
```

```
        String RegEmail = request.getParameter("email");
        String RegPass = request.getParameter("password");
String RegPass2 = request.getParameter("password2");
```

First, it will check for filling all fields then check for compatibility of group and coordinator passwords. As following:

```
// if compatible group's passwords
if ( gpass.equals(gpass2))
{// if compatible coordinator's password
        if(RegPass.equals(RegPass2) ){
```

After that it will check for availability of suggested group name. As following:

```
String vgroup= null ;
rs = selectQuery("SELECT * "+"FROM project "+
"WHERE group_name=\""+ group+ "\"");
while(rs.next()){ vgroup=rs.getString("group_name"); }
```

If group name is available and no error occurs registration completed by accessing appropriate tables of CFGP database. As following :

```
if (vgroup==null)
{       //add group
            query =  "insert into project values(?,?) " ;
        PreparedStatement stmtt=        Connection.prepareStatement(query);
                stmtt.setString( 1, group);
```

```
stmtt.setString(2, gpass);
int  resultSet = stmtt.executeUpdate();
query =  "insert into coordinator values(?,?,?,?,?) " ;
stmtt = Connection.prepareStatement(query);
stmtt.setString( 1, RegUser);
stmtt.setString(2, RegPass);
stmtt.setString( 3, RegEmail);
stmtt.setString( 4, "coordinator");
stmtt.setString( 5,group);
resultSet = stmtt.executeUpdate();
```

Finally, group implementation completes by creation of group shared folder and group chat rooms:

a. Creation of group shared folder on server disk, As following:

```
//create shared folders
    String server_path = "C:/";
    String Dirspath = server_path+"shared folders/"+group;
    if(new File(Dirspath+"/Team work").mkdirs())
 if(new File(Dirspath+"/Supervisor/from").mkdirs())
 if(new File(Dirspath+"/Supervisor/to").mkdirs())
response.sendRedirect("/info.jsp?action=regcomplete");
```

b. Creation of group's discussin room (private and public) as servlet web page. As following:

```
addNewRoom(rooms, roomName, roomDescr);
String ProomName=roomName+"-private";
addNewRoom(rooms, ProomName, roomDescr);
```

In addNewRoom function, set all attribute of room then add it to room list as a following :

```
ChatRoom room = new ChatRoom(roomName, roomDescr);
room.setMaximumNoOfMessages(maxMessages);
rooms.addRoom(room);
```

To make the room available after registration, it must be saved in chat property file as a following:

```
props.put(roomName, roomDescr);
String path = "/WEBINF/"+getServletContext().getInitParameter("chatpropertyfile");
String realPath = getServletContext().getRealPath(path);
OutputStream fos = new FileOutputStream(realPath);
props.store(fos, "List of Rooms");
fos.close();
```

Property file is a flexible text file that ease the reading and the writing from\to file, as following statements shown below:

```
#List of Rooms
StartUp=Startup chat room. Chatter is added to this after he logs in.
Group1=with different members.
```

**2-Join to group.**

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

❖ **At client side:**

At first and through browser, supervisor or member fill join to group form with required group's name and password see figure (5.4).

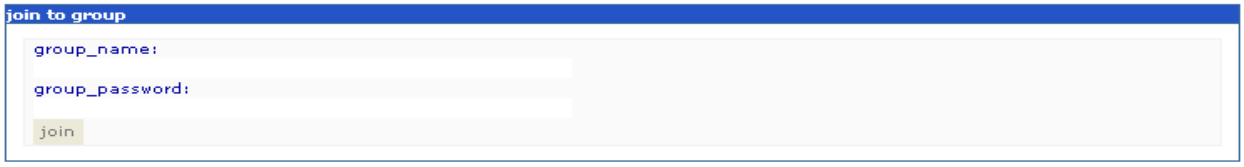



**Figure 5.4: join to group.**

These group's authentication data submitted to servlet file at server in order for validation. However , if these data were accepted server forward register to fill registration form presented in JSP file with applicable member information which will be processed by registration handler servlet as requests. However , E-mail format is validated by JavaScript function on client side using regular expression concepts.

❖ **At server side:**

At first registration servlet handler get http requests ( group's data) from client(register) . It will perform some queries on CFGP database to validate presented data (see group authentication). If authentication proceeds with wrong group's data servlet file will forward register to JSP file show his\her error explanation message .However if group data authenticated, server will forward to other servlet file which will insure at first that all fields filled with acceptable data. Then, the program must verify that there is no supervisor registered in group if a team member type was supervisor. As shown below:

```
if (   ( utype.equals("supervisor") )  ){
rs =  selectQuery("SELECT * "+"FROM supervisor "+
                  "WHERE Group_name=\""+ group+ "\"");
while(rs.next()){  sfound= rs.getString("type");}}
//end of scan for supervisor }
```

Then check if a name registered for any one existed in same group, as shown in following fragment of registration code.

```
// if no previous supervisor , ok search for existing name
if ( sfound!=null){
rs = selectQuery("SELECT * FROM member WHERE member_name =\""+
RegUser + "\" AND Group_name=\""  + group+ "\"");
while(rs.next()){
DBUsername = rs.getString("member_name");                    }
if(DBUsername == null){
rs = selectQuery("SELECT * FROM coordinator WHERE coname=\""+ RegUser
+"\" AND Group_name=\""  + group+ "\"");
while(rs.next()){
DBUsername = rs.getString("coname");}
if(  (DBUsername == null ) ){
rs = selectQuery("SELECT * FROM supervisor WHERE sname=\""+ RegUser +
        "\" AND Group_name=\""  + group+ "\"");
while(rs.next()){
        DBUsername = rs.getString("sname");}}      }
```

Registration will be completed by adding register information into appropriate tables of CFGP database through following query. As shown in following fragment of registration code.

```
// if no prevoiouse same name check for compatible passwords
if(DBUsername == null){
if(RegPass.equals(RegPass2) ){
```

```
query =  "insert into "+ utype+" values(?,?,?,?,?) " ;
PreparedStatement stmtt = Connection.prepareStatement(query);
                stmtt.setString( 1, RegUser);
                stmtt.setString(2, RegPass);
                stmtt.setString( 3, RegEmail);
                stmtt.setString( 4, utype);
                stmtt.setString( 5,group);
              int  resultSet = stmtt.executeUpdate();
```

If some errors occur during these validations, type of error submitted to info.jsp file which will view error messages. As shown in following fragment of registration code.

```
response.sendRedirect("/info.jsp?action=regcomplete");
}//closed if compatible pass
else// NO COMATIBLE PASS{
response.sendRedirect("/info.jsp?action=wrongpass");}
}// closed if not null
else //there is previouse name{
response.sendRedirect("/info.jsp?action=userexists");}
} //end if no supervisor
else//there is supervisor in the indicated group{
response.sendRedirect("/info.jsp?action=supervisorexists"); }}
```

- **View error messages :**

All registration errors (see figure 5.5) have been got at registration validation process redirect to JSP file which will view explanation error message depends on parameter indicates that error type like following fragment of code**:**

```
else//there is supervisor {

response.sendRedirect("/info.jsp?action=supervisorexists");}
```

**Figure 5.5: a supervisor existed in a group.**

- **Group authentication :**

In order to join to existed group, team member must authenticate group information which will be posted with sessions into testgroup.java

This will validate group's information by accessing CFGP database and making search query to insure existence of group. As shown in following fragment of authentication registration code.

```
HttpSession session = request.getSession(true);

String vgname=request.getParameter("group");

String vpass=request.getParameter("gpass");

 stmt= connection.createStatement();

String query="SELECT * FROM project WHERE group_name="'+ vgname +'"and gpass="'+vpass+'"";

ResultSet rs = stmt.executeQuery(query);

while(rs.next()){

                result= rs.getString("group_name");  }
```

Then, depend on validation result it will forward to registration form with data in session or view an explanation of error. As shown in following fragment of registration code.

```
if(result==null){
response.sendRedirect("/wrong.jsp");
}else
//redirect into registeration form
{ session.setAttribute("group" , result);
response.sendRedirect("/reg_gmember.jsp");}
```

- **Password Encryption:**

Data security has become important as more information is in a digital form. We need security services to protect data from unauthorized access. To solve this problem,

Sun Microsystem, the creator of Java, has released new cryptographic packages that are included with Java 2 Standard Edition version 1.4. The packages are called Java Cryptography Extension (JCE) support wide range of cryptographic services including digital signatures, message digests, ciphers (symmetric and asymmetric), message authentication codes, key generators and key factories [29] .

By using (JCE), we applied shared key encryption on passwords of CFGP database.

We needed some packages like **import java.security.*** which provides the classes and interfaces for the security framework and **import javax.crypto**.* which Provides the classes and interfaces for cryptographic operations.

We give a code sample from Registration process of how the JCE works.

1- Define parameters:

```
String plaintext = userPassword;
String mykey = "1234567887654321"; // needed to be exactly 16 bytes
```

```
String myiv = "0123456789abcdef"; // must be 16 bytes for AES
```

2- We define a key using SecretKeySpec class, and we need to specific what the key is for. In this sample is for AES, in which the key has to be 16 bytes long. We also define the IV since we will use AES in CBC mode. The JCE requires the IV to be exactly 16 bytes long.

```
// define a key
SecretKeySpec keyspec = new SecretKeySpec(mykey.getBytes(), "AES");
// define IV
 IvParameterSpec ivspec = new IvParameterSpec(myiv.getBytes());
```

3- We need to instantiate the cipher and specify the cipher type, the cipher mode and the padding mode. We use backslash (/) to specify them into one string. In this sample, we use AES in CBC mode and use PKCS#5 padding which is a common standard for data padding.

```
// set the cipher to desired operations
 Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

4- We use init method to set whether it is for encryption or decryption, and also to set the key and IV.

```
// initiate the cipher
 cipher.init(Cipher.ENCRYPT_MODE, keyspec, ivspec);
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

5- We encrypt the plaintext using doFinal method. The doFinal method will pad the data if it is less than the specified block size. The encryption takes bytes as input, and also produces bytes as output.

```
// Encrypt
byte [] encrypted = cipher.doFinal(plaintext.getBytes());
```

6- To make the encrypted text readable, we use Base64 encoding. Use Base64Encoder class to encode encrypted bytes [30][31].

```
// Encode
String encoded = new sun.misc.BASE64Encoder().encode(encrypted);
```

- **Delete Group**

❖ **At client side:**

Delete group assumed to be processed by group's coordinator. From browser, coordinator press delete group button see figure (5.6). This process submitted as a request to delete group servlet handler at server side to process them.

**Figure 5.6: Delete Group.**

❖ **At server side:**

Delete group handler servlet gets data as requests from delete group button at client side. Then, connecting to CFGP database and performing queries .If no error occurs, delete group completed by display a home page of CFGP web based tool.

To delete a group, a query sends to table **project** to delete the group,

```
String query = "DELETE FROM project WHERE group_name=\"" + sessionGroup +
"\"" ;

PreparedStatement stmtt = db.Prequery(query);

int  resultSet = stmtt.executeUpdate();
```

Then, member's data send to **list** table by a delete query to delete member's data from contact list (see contact list).

```
query = "DELETE FROM list WHERE name=\"" + sessionUsername + "\" and type
=\"" + sessionType + "\" and Group_name=\"" + sessionGroup + "\"";

 stmtt = db.Prequery(query);

 resultSet = stmtt.executeUpdate();
```

Finally, remove session attributes as we will see in log out.

- **Logging in:**

After registration completed, member is able to login to the CFGP web based tool.

❖ **At client side:**

Each time the member want to use CFGP web based tool a login form should be filled with the member's login required data (see figure 5.7).These data submitted as a request to login servlet handler at server side to process them. But invalid login cause the tool to show an error message.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

**Figure 5.7: login form.**

- **At server side:**

  At first, login servlet handler get http requests (member's data) from login form at client side. It will perform some queries on CFGP database to validate presented member's data. If authentication proceeds with wrong member's data, servlet file will forward login to JSP file show his\her error explanation message. If no error occurs, login completed by display group page as a home page of CFGP web based tool.

  Authentication from member's data implemented by send queries to **member, coordinator and supervisor tables**.

```
ResultSet rs = db.selectQuery("SELECT * "+"FROM member "+
"WHERE name =\"" + loginUser + "\" AND group_name =\"" + loginGroup + "\"
AND pass =\"" + loginPass + "\"");
```

```
ResultSet rs2 = db.selectQuery("SELECT * "+"FROM coordinator "+
"WHERE name=\"" + loginUser + "\" AND group_name =\"" + loginGroup + "\"
AND pass =\"" + loginPass + "\" ");
```

```
ResultSet rs3 = db.selectQuery("SELECT * "+"FROM supervisor "+
"WHERE name=\"" + loginUser + "\" AND group_name =\"" + loginGroup + "\"
AND pass =\"" + loginPass + "\" ");
```

If member's data exist in one of previous tables, the attributes of session are initialized:

```
HttpSession session = request.getSession(true);
        session.setAttribute("username",loginUser);
        session.setAttribute("password",password);
        session.setAttribute("type",type);
        session.setAttribute("group",loginGroup);
```

Then, member's data send to **list** table by an insert query to show in contact list (see contact list).

```
String query ="insert into list (name,type,Group_name)values(?,?,?)";
PreparedStatement stmtt = db.Prequery(query);
                stmtt.setString( 1, loginUser );
                stmtt.setString( 2, type );
                stmtt.setString( 3,loginGroup );
        // Send query to database.
        int  resultSet = stmtt.executeUpdate();
```

- **Log out:**

❖ **At client side:**

From browser, member press log out link see figure (5.8). This process submitted as a request to log out servlet handler at server side to process them.

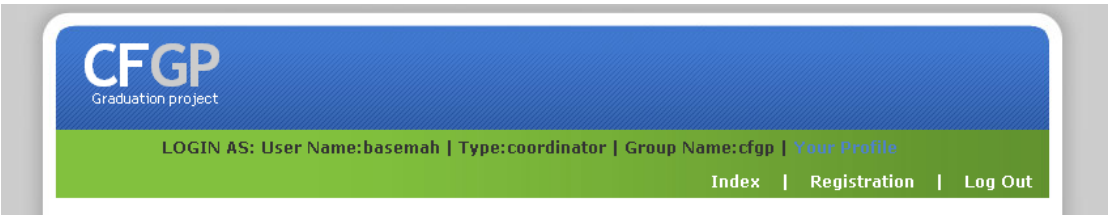

CFGP
Graduation project
LOGIN AS: User Name:basemah | Type:coordinator | Group Name:cfgp | Your Profile
Index | Registration | Log Out

**Figure 5.8: member information.**

- **At server side:**

  Log out handler servlet gets data as requests from log out link at client side. Then, remove session attributes.

```
session.removeAttribute("username");
session.removeAttribute("password");
session.removeAttribute("type");
session.removeAttribute("group")
```

After that, member's data send to **list** table by a delete query to delete member's data from contact list (see contact list).

```
String query = "DELETE FROM list WHERE name=\"" + sessionUsername + "\" and type =\"" + sessionType + "\" and Group_name=\"" + sessionGroup + "\""
PreparedStatement stmtt = db.Prequery(query);
// Send query to database.
int  resultSet = stmtt.executeUpdate();
```

If no error occurs, log out completed by display log in form as a home page of CFGP web based tool.

### 5.2.3. Discussion Room:

Firstly, the rooms list is created when the server is started and where a new group is registered. To inter discussion room we have to read chat properties file which includes the names of groups and their descriptions as a following:

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
path="/WEBINF/"+getServletContext().getInitParameter("chatpropertyfile");
String realPath;
realPath = getServletContext().getRealPath(path);
if (realPath != null)
  {InputStream fis = new FileInputStream(realPath);
     props = new Properties();
          props.load(fis);
          Enumeration keys = props.keys();
          while (keys.hasMoreElements())
            {       String roomName = (String)keys.nextElement();
                    String roomDescr = (String)props.getProperty(roomName);
                    addNewRoom(rooms, roomName, roomDescr);
                    String ProomName=roomName+"-private";
                    addNewRoom(rooms, ProomName, roomDescr); }
          fis.close();
          getServletContext().setAttribute("chatroomlist", rooms);
          System.err.println("Room List Created");  }
```

When the member wants to enter his group discussion room, he clicks on open or private discussion room icon in group page, where the private icon is hidden for supervisor.

These icons are linked to **LoginServlet**.java that takes the name and the type from the session and creates a new chatter that initially added in StartUp room as a following:

```
ChatRoom chatRoom = roomlist.getRoom("StartUp");
chatter = new Chatter(name, type, new java.util.Date().getTime());
chatRoom.addChatter(chatter);
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

Then, it redirects to **start.jsp**. This file takes a group member session and opens the room that has a same group name. Then, it adds a member as a chatter in this room and set the time that the chatter enters this room as a following:

```
String roomname = (String)session.getAttribute("group");

ChatRoom chatRoom = roomlist.getRoom(roomname);

chatRoom.addChatter(chatter);

chatter.setEnteredInRoomAt(new java.util.Date().getTime());
```

Then, it redirects to displayed page. There are three parts:

* First part in the left top page for displaying the messages.

* The second in the right top page for displaying the contact list in the room.

* The third in the bottom for entering the message and clicking the send button that display messages in first part as in Figure(5.9).

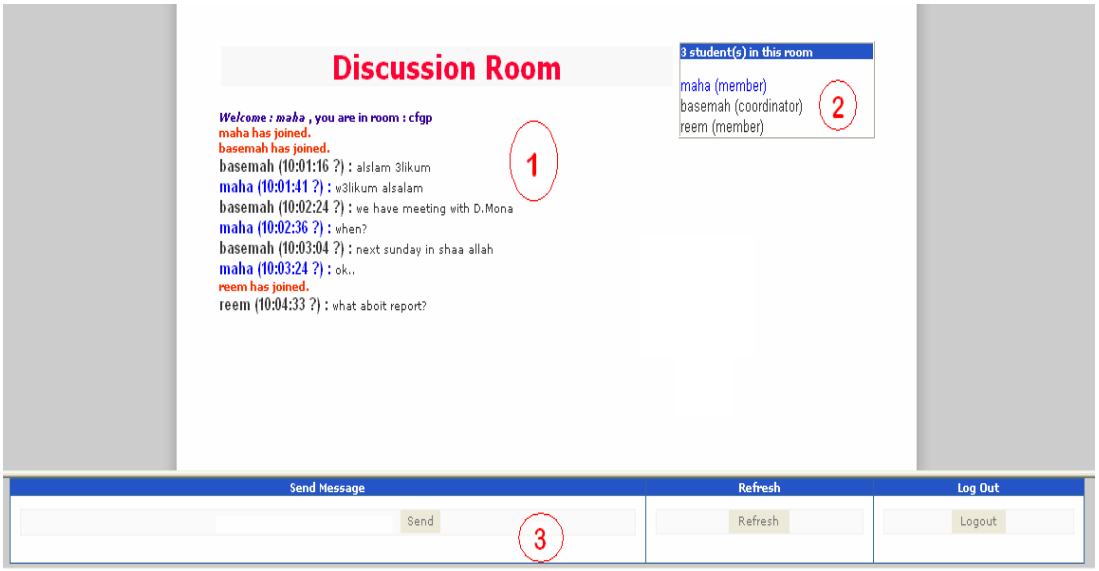



**Figure 5.9: Discussion Room.**

There is also two buttons: first for log out and the second for refresh page. The refreshing automatically is done after 7 seconds as a default and is set in the **web.xml** as a following:

```
<context-param>
<param-name>refreshAfter</param-name>
<!—
Add time in seconds after which user's browser window will automatically refresh to display new messages.
Default is 7 seconds.
-->
        <param-value>7</param-value>
        </context-param>
```

But if the user wants to refresh before 7 seconds, he clicks on refresh button.

In this feature, we need four data structures presented as classes as a following:

1- ChatRoomList contains the following:

```
/*** Stores all the ChatRoom objects*/
        private Map roomList;
```

In this class, there are two synchronized methods, as a following:

```
        /*** adds new chat room object to a list of Rooms.
        * @param room ChatRoom object
```

```
        * @return void*/
        public synchronized void addRoom(ChatRoom room)
        {roomList.put(room.getName(), room);}
        /*** Used to remove a ChatRoom object from the list of ChatRooms.
        * @param name is a String object is the name of the
        * room to be removed from the list of rooms.
        * @return void*/
        public synchronized void removeRoom(String name)
        {roomList.remove(name);}
```

Here, we need acquire-release protocol by using these synchronized methods for concurrency access to shared object (roomList).

2- ChatRoom contains the following:

```
    /** used to stroe name of the room*/
        private String name = null;
        /** used to store description of the room*/
        private String description = null;

        /** Map to store Chatter objects*/
        private Map chatters = new HashMap();
        /** Linked list to store Message object*/
        private List messages = new LinkedList();
        /** Used to set the maximum no of messages*/
        private int messages_size = 25;
```

In this class, there are two synchronized methods, as a following:

```
/*** adds a Chatter object to list of Chatters
* @param chatter Chatter object
* @return void*/
public synchronized void addChatter(Chatter chatter)
{chatters.put(chatter.getName(), chatter);}
/*** removes a Chatter object from list of Chatters
* @param chatterName name of the chatter.
* @return void*/
public synchronized Object removeChatter(String chatterName)
{return chatters.remove(chatterName);}
```

Here, also we need acquire-release protocol for concurrency access to shared object(chatters).

3- chatter contains the following:

```
/* * name of the chatter
* type of the chatter */
private String name;
private String type;
```

4- Message that represents a message sends by member, contains the following:

```
/*** String used to store the name of a chatter*/
private String chatterName = null;
/*** String containing message*/
private String message = null;
```

```
/*** long containing the time when message was delivered*/
        private long timeStamp;
```

In this feature, we use servlet,jsp and javascript as techniques for implementation. We use also http as application protocol. We also need acquire-release protocols for synchronization by using synchronized keyword.

The effect of making these methods ***synchronized*** has two effects:

- First, it is not possible for two invocations of synchronized methods on the same object to interleave. When one thread is executing a synchronized method for an object, all other threads that invoke synchronized methods for the same object block (suspend execution) until the first thread is done with the object to ensure that can be executed by only one thread at a time, to prevent data loss or corruption. This one thread running at a time property is called ***mutual exclusion***.
- Second, when a synchronized method exits, it automatically establishes a "happens-before" relationship with any subsequent invocation of a synchronized method for the same object. "***Happens-before"*** relation on memory operations such as reads and writes of shared variables, that the results of a write by one thread are guaranteed to be visible to a read by another thread only if the write operation happens-before the read operation

### 5.2.4. Shared Folders:

Sharing folders allows remote web-based file access and manipulation with features of uploading and downloading. User can copy, move and delete files. Text files can be created and edited. Groups of files and folders can be downloaded as a single zip file. All of this features done over HTTP protocol by **FileSharing.jsp** page imports deferent packages, some of them are:

**Table 5. 2 CFGP shared folder java used packages**

| *Package* | *Description* | *Some Interfaces and Classes used* |
|---|---|---|

| | | |
|---|---|---|
| *Java.util* | Contains framework for representing and manipulating collections | **Hashtable**: implements a hashtable, which maps keys to values. |
| *Java.net* | Provides the classes for implementing networking applications. | **URLDecoder:** Utility class for HTML form decoding. |
| | | **URLEncoder**: Utility class for HTML form encoding. |
| *Java.util.zip* | Provides classes for reading and writing the standard ZIP and GZIP file formats.<br>Also includes classes for compress and decompress data using the DEFLATE compression algorithm,<br>which is used by the ZIP and GZIP file formats. | **ZipEntry:** used to represent a ZIP file entry. |
| | | **ZipInputStream**: implements an input stream filter for reading files in the ZIP file format. |
| | | **ZipOutputStream**: implements an output stream filter for writing files in the ZIP file format. |
| *Java.io* | Provides for system input and output through data streams, serialization and the file system. | **File** An abstract representation of file and directory pathnames. |

**- Class java.util.Hashtable :**

This class implements a hashtable, which maps keys to values. Any non-null object can be used as a key or as a value.

An instance of Hashtable has parameters that affect its performance:

- Flexible size, initial capacity is simply the capacity at the time the hash table is create can automatically increased when need, and a single bucket can stores multiple entries.
- Fail-Fast: behavior used to detect bugs. If the Hashtable is structurally modified at any time after is created through remove or add methods, it will throw an exception. Thus, fails quickly and cleanly, rather than risking arbitrary.
- Three basics operation done:
    - Removes the key (and its corresponding value) from this hashtable. This method does nothing if the key is not in the hashtable.
    - The value can be retrieved by calling the get method with a key that is equal to the original key.
    - Calling the put method maps the specified key to the specified value in this hashtable. Neither the key nor the value can be null.

**- Class Java.net.URLDecoder:**

This class contains static methods for decoding a String from the application/x-www-form-urlencoded MIME format.

**- Class java.net.URLEncoder:**

This class contains static methods for converting a String to the application/x-www-form-urlencoded MIME format.

The following rules are applied in the conversion:

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

- The alphanumeric characters "a" through "z", "A" through "Z" and "0" through "9" remain the same. Also, the special characters ".", "-", "*", and "_" remain the same.
- The plus sign "+" is converted into a space character " " for decoding and , The space character " " is converted into a plus sign "+" for encoding.
- All other characters converted into one or more bytes using some encoding scheme

## Shared Folder Implementation:

Shared folder holds all group folders. Group folder named with their group name. Sharing folder can restrict file accessing and manipulation by using restriction, that means the user is allowed to access only directories specified , which are wrote in RESTRICT_PATH variable, that useful in two things : security for groups work and privacy for the group members beside the supervisor, where the subfolder (Team work) will not shown by the supervisor, the supervisor will show only the subfolder (Supervisor). This done as follow:

```
//After get: Group(group name) , Type (user type) from session
public  static String RESTRICT_PATH ="C:\\Shared Folder\\"+Group;
if ( Type.equalsIgnoreCase("supervisor"))
        RESTRICT_PATH = "C:\\Shared Folder\\"+Group+"\\Supervisor";
```

After that, all operations on files request by user , the web-path of file checked if allowed or not by calling method **isAllowed** , take the file path, then compare the beginning with the **RESTRICT_PATH** , if its equal return allow or return not if not equal, .

```
static boolean isAllowed(File path) throws IOException
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
{ if ( path!=null &&
        path.getCanonicalPath().startsWith(RESTRICT_PATH))
        return true;
       return false;}
```

For example:

Path1="C:\Shared Folder\CFGP\Team Work\Job\note.txt"

Path2="C:\Shared Folder\ note.txt"

RESTRICT_PATH="C:\Shared Folder\CFGP"

Path1 is allowed while path2 is not.

Deferent classes and method written to facilitate action performed. Upload needed information like: file total size , time estimate , time elapsed , etc , all represented and calculate by an object of , as follow.

```
public class UplInfo. { /*... data members and methods … */ }
```

File information like : file name, type, contents ,etc represented by an object of, as follow.

```
public class FileInfo { /*... data members and methods … */ }
```

Also, a method takes the parent directory name and the file name ,then return a URL full path of the file as String.As following .

```
public String getFileName(String dir, String fileName);
```

Know that, all operations within shared folder (accesses or manipulates) based on HTTP URL, as follow.

```
//Get the current browsing directory
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
        request.setAttribute("dir", request.getParameter("dir"));
// The browser_name variable is used to keep track of the URI
// of the jsp file itself.  It is used in all link-backs.
        final String browser_name = request.getRequestURI();
```

and the URL for all actions written as:

```
<a href=\"" + browser_name + "?sort=" + sortMode + "&amp;dir="
        + URLEncoder.encode(f.getParent()) + "\">" + "[FileName]</a>"
```

Where:

- **browserLink:** web-path to FileSharing.jsp
- **sortMode :** files and folders view mode sort:

  1=Filename, 2=Size, 3=Date, 4=Type,

  The default sorting method is by Name

- **URLEncoder.encode(f.getParent()):** file parent directory .

Because the appearance of a Web page may different between browsers and user may used deferent browsers, and according to the client browser the deal with the request may be differentiate , we need to check the browser type by JavaScript function:

```
function CheckBrowser(){
    if (b == 0){
        if (window.opera) OP = 1;
        if(document.getElementById) DOM = 1; // Moz or Netscape
```

```
if(document.all && !OP) MS = 1; // Micro$oft
        b = 1;}}
```

For operation like : open and show directory contents, or select group of file to download, delete , move or copy, we need to store the files name and information this done perfectly with an object of Vector expand files list from top-bottom, starting with the directories if have so set **Incldir=true** , or **Incldir=false** if there are no dircitories . And after fill an array of string with the files names call the method.

```
static Vector expandFileList(String[] files, boolean inclDirs) {
Vector v = new Vector();
if (files == null) return v;
for (int i = 0; i < files.length; i++)
        v.add(new File(URLDecoder.decode(files[i])));
for (int i = 0; i < v.size(); i++) {
        File f = (File) v.get(i);
        if (f.isDirectory()) {
                File[] fs = f.listFiles();
                for (int n = 0; n < fs.length; n++)
                        v.add(fs[n]);
                if (!inclDirs) {
                        v.remove(i);
                        i--;}}}return v;}
```

Also, for basics operations like move, copy, create file/dir, need the absolute path (where the new file/dir will store, An absolute pathname is complete in that no other information is required in order to locate the file that it denotes) get from follow.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
static String getDir(String dir, String name) {
        File mv = new File(name);
        if (!mv.isAbsolute())
                return dir + name;
        else return name;}
```

Dealing with stream write and read need copies all data from input stream to the output stream or verse versa, a copy buffer used as follow:

```
static void copyStreams(InputStream in, OutputStream out, byte[] buffer)throws
IOException { int b;
                while ((b = in.read(buffer)) != -1)
                        out.write(buffer, 0, b);
                in.close();    out.close();}
```

Usually, when a HTTP Response is made from the web server to a client, it includes extra information in the HTTP HEADER. These HTTP header lines contain many useful information , like:

**Table 5. 3 HTTP Headers**

| ***Header Name*** | ***Description*** |
|---|---|
| HTTP Status | Contains an integer that describes how client is to process the HTTP response. |
| Pragma | Used by servers to tell HTTP/1.0 clients not to cache, and to send extra data to client. |
| Cache-Control | Tells the client to set cache properties for the current HTTP |

| | |
|---|---|
| | Response. |
| Connection | Tells client to hold or abandon connection after HTTP data transfer. |
| Content-Encoding | Tells client what the HTTP Body contains. |
| Content-Language | Tells client what language is used in the HTTP Body. |
| Content-Length | Tells the client the exact size of the HTTP Body. |
| Date | Tells the client the current GMT at server. |
| Expires | Tells the client the GMT data when this page should be considered old and should be refreshed. |
| Location | Tells the client another URI as in a redirect. |
| Set-Cookie | Tells the client to create one or more cookies. |

This information can retrieve using method getLine(), as follow.

```
private synchronized String getLine(ServletInputStream sis) throws IOException {
        byte b[] = new byte[1024];
        int read = sis.readLine(b, 0, b.length), index;
        String line = null;
        if (read != -1) {
                line = new String(b, 0, read);
                if ((index = line.indexOf('\n')) >= 0)
                line = line.substring(0, index - 1);}
        return line;}
```

Method getLine() called by methods of class, as follow.

```
public class HttpMultiPartParser { /*.. data members and methods */ }
```

This class has a method that create a specific hashtable of files to upload , as follow.

```
HttpMultiPartParser parser = new HttpMultiPartParser();

int bstart = request.getContentType().lastIndexOf("oundary=");

String bound = request.getContentType().substring(bstart + 8);

int clength = request.getContentLength();

Hashtable ht = parser.processData(request.getInputStream(), bound, tempdir,
clength);
```

## Shared Folder Basic Tasks Implementation:

- **Upload File:**

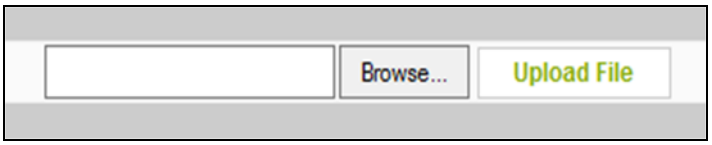



**Figure 5.10: upload file.**

❖ **At client side:**

User choose a file to upload by clicking on ( Browse…) button , then submit request by clicking on (Upload File) button see figure 5.10 .

```
<input type="file" class="textfield" onKeypress="event.cancelBubble=true;"
name="myFile">

<input title="Upload selected file to the current working directory" type="Submit"
class="button" name="Submit" value="<%=UPLOAD_FILES%>"

onClick="javascript:popUp('<%= browser_name%>')">
```

❖ **At server side:**

After open a file chooser window (allow user to choose file want to upload).See figure 5.11.

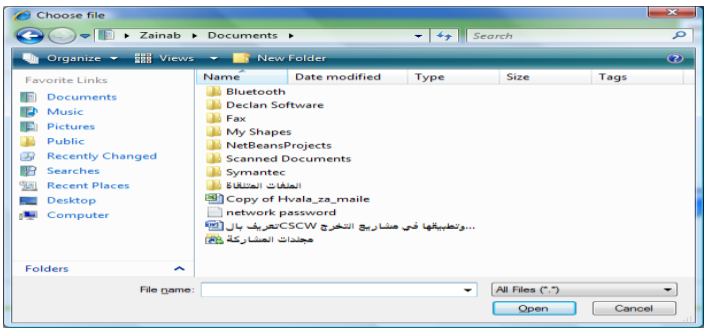



**Figure 5.11: Choose file.**

Then, request send to JavaScript function retrieve file name and open upload monitor, as follow.

```
function popUp(URL){

fname = document.getElementsByName("myFile")[0].value;

if (fname != "")

window.open(URL+"?first&uplMonitor="+encodeURIComponent(fname),"","width=400,hei
ght=150,resizable=yes,depend=yes")}
```

Upload monitor is a graphical window show the upload of the file, as follow.

```
<META HTTP-EQUIV="Refresh"
CONTENT="<%=UPLOAD_MONITOR_REFRESH%>;URL=<%=browser_name
%>?uplMonitor=<%=URLEncoder.encode(fname)%>">

</head>

<body>

<b>Upload of <%=fname%></b><br><br>

<%=convertFileSize(info.currSize)%> from <%=convertFileSize(info.totalSize)%>
```

```
(<%=info.getPercent()%> %) uploaded (Speed: <%=info.getUprate()%>).<br>
Time: <%=info.getTimeElapsed()%> from <%=info.getTimeEstimated()%>
</body>
```

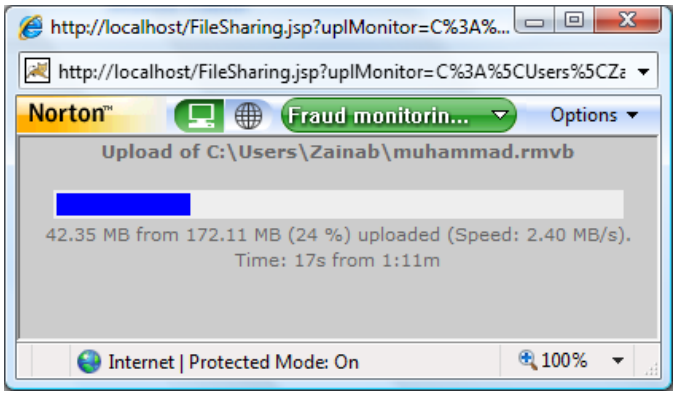



**Figure 5.12: :file is uploading**

Where the time the window refresh in second set at, as follow.

```
private static final int UPLOAD_MONITOR_REFRESH = 3
```

Know that, UploadMonitor class structure is:

```
public static class UploadMonitor {
        static Hashtable uploadTable = new Hashtable();
        static void set(String fName, UplInfo info)
{                       uploadTable.put(fName, info);   }
        static void remove(String fName)
{               uploadTable.remove(fName);              }
        static UplInfo getInfo(String fName)
{       UplInfo info = (UplInfo) uploadTable.get(fName);
                return info;                                    }}
```

When UploadMonitor opened, the upload operation start, this operation done by method as we say before:

```
public Hashtable processData(ServletInputStream is, String boundary, String saveInDir,int
clength) throws IllegalArgumentException, IOException;
```

Upload done as follow:

```
int ONE_MB = 1024 * 1;
UplInfo uplInfo = new UplInfo(request.getContentLength());
String path = getFileName(saveInDir,fileInfo.clientFileName);
OutputStream os = new FileOutputStream(path);
byte previousLine[] = new byte[2 * ONE_MB];
byte temp[] = null;
byte currentLine[] = new byte[2 * ONE_MB];
int read, read3;
while (readingContent) {
if ((read3=is.readLine(currentLine,0,currentLine.length))==-1){
        line = null;
                uplInfo.aborted = true;
                break;}
        if (compareBoundary(boundary, currentLine)) {
                os.write(previousLine, 0, read - 2);
                line = new String(currentLine, 0, read3);
                break;}
        else {  os.write(previousLine, 0, read);
                uplInfo.currSize += read;
                temp = currentLine;
                currentLine = previousLine;
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
                    previousLine = temp;
                    read = read3;}//end while
os.flush();
os.close();
uplInfo.currSize = uplInfo.totalSize;
```

- **Download Single File:**

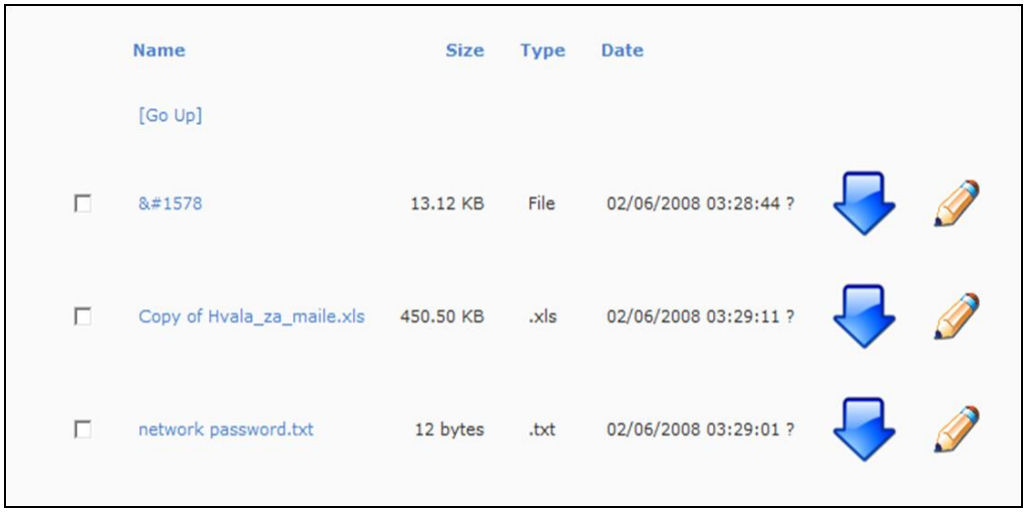



**Figure 5.13: download file.**

❖ **At client side**:

User click on the image of the file want to download it.

```
href+"downfile="+name+"\"><img src=\"download.png\"
        alt=\"Download\"></a>";
```

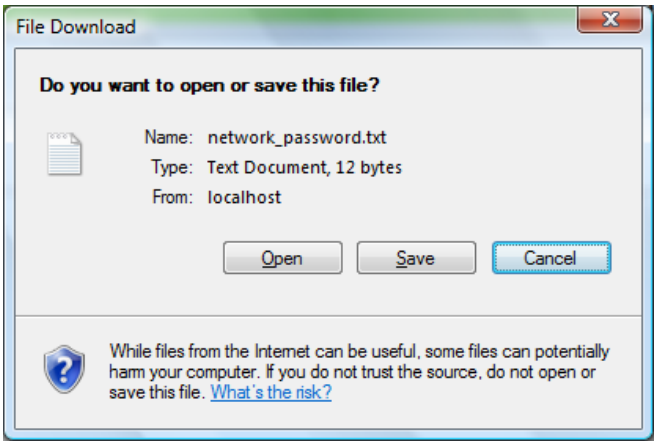

File Download
Do you want to open or save this file?
Name: network_password.txt
Type: Text Document, 12 bytes
From: localhost
Open
Save
Cancel
While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

**Figure 5.14: open or save a file.**

- **At server side:**

```
String filePath = request.getParameter("downfile");
        File f = new File(filePath);
        if (f.exists() && f.canRead()) {
                response.setContentType("application/octet-stream");
                response.setHeader("Content-Disposition",
                "attachment;filename=\"" +  f.getName()+ "\"");
                response.setContentLength((int) f.length());
                BufferedInputStream fileInput = new
                        BufferedInputStream(new FileInputStream(f));
                byte buffer[] = new byte[8 * 1024];
                out.clearBuffer();
                OutputStream out_s = new Writer2Stream(out);
                copyStreamsWithoutClose(fileInput, out_s, buffer);
                fileInput.close();
                out_s.flush();
                nohtml = true;
                dir_view = false;}
```

- **Download as zip:**

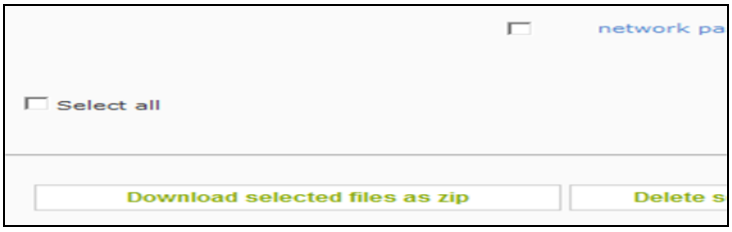



**Figure 5.15: download as zip.**

❖ **At client side:**

Checked the files want to download then click download button.

❖ **At server side:**

```
if ((request.getParameter("Submit") != null) &&
(request.getParameter("Submit").equals(SAVE_AS_ZIP))) {

Vector v = expandFileList

        (request.getParameterValues("selfile"), false);

                File dir_file=new File("" +request.getAttribute("dir"));

                int dir_l = dir_file.getAbsolutePath().length();

                out.clearBuffer();

                ZipOutputStream zipout = new ZipOutputStream(new

                Writer2Stream(out));

                zipout.setComment("Created by CFGP web based tool");

                zipout.setLevel(COMPRESSION_LEVEL);

                for (int i = 0; i < v.size(); i++) {

                        File f = (File) v.get(i);

                        if (f.canRead()) {

                                zipout.putNextEntry(new

                         ZipEntry(f.getAbsolutePath().substring(
```

```
                    dir_l + 1)));
                    BufferedInputStream fr = new
                            BufferedInputStream(new
                    FileInputStream(f));
                    byte buffer[] = new byte[0xffff];
                    copyStreamsWithoutClose(fr, zipout, buffer);
                    fr.close();
                    zipout.closeEntry();}}
            zipout.finish();
            out.flush();
            nohtml = true;
            dir_view = false;}}
```

- **Delete file:**

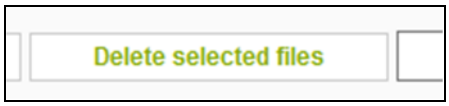



**Figure 5.16: delete file.**

❖ **At client side:**

Checked the files want to delete then click delete button.

❖ **At server side:**

```
Vector v= expandFileList(request.getParameterValues("selfile"),true);
//delete backwards
for (int i = v.size() - 1; i >= 0; i--) {
File f = (File) v.get(i);
f.delete();}
```

## Other optional tasks:

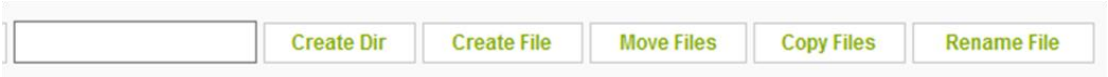



**Figure 5.17: optional tasks.**

- **Create Dir:**

❖ **At client side:**

Write Dir name at text filed, then click Create Dir button.

❖ **At server side:**

```
if (new File(new_dir).mkdirs())
        request.setAttribute("message", "Directory created");
```

- **Create File:**

❖ **At client side:**

Write file name at text filed,then click Create File button.

❖ **At server side:**

```
if (new File(new_file).createNewFile())
request.setAttribute("message","File created");
```

- **Move File:**

❖ **At client side:**

Select the files want to move, write new path at the text field,then click move button.

❖ **At server side:**

```
Vector v=expandFileList(request.getParameterValues("selfile"),true);
for (int i = v.size() - 1; i >= 0; i--) {
File f = (File) v.get(i);
f.renameTo(new File(new_dir+f.getAbsolutePath().substring(dir.length())));}
```

- **Copy File:**

❖ **At client side:**

Select the files want to copies, write the path at the text field, then click Copy button.

❖ **At server side:**

```
Vector v=expandFileList(request.getParameterValues("selfile"),true);
byte buffer[] = new byte[0xffff];
for (int i = 0; i < v.size(); i++) {
        File f_old = (File) v.get(i);
        File f_new = new File(new_dir +
       f_old.getAbsolutePath().substring(dir.length()));
   if (f_old.isDirectory())
       f_new.mkdirs();
   else if (!f_new.exists())
      copyStreams(new FileInputStream(f_old), new
                FileOutputStream(f_new), buffer);              }
```

- **Rename File:**

❖ **At client side:**

Select only one file or dir, write the new name in the text field, click Rename button.

❖ **At server side:**

```
Vector v=expandFileList(request.getParameterValues("selfile"),true);
```

```
File f = (File) v.get(0);
if (f.renameTo(new File(new_file.trim())))
request.setAttribute("message", "Renamed file "
                        + ((File) v.get(0)).getName() + " to " + new_file);
```

- **Edit:**

**Figure 5.18: Edit file button.**

❖ **At client side:**

Click on the image of the ***text*** file want to edit.

```
href+"editfile="+name+"\"><img src=\"edit.gif\" alt=\"Edit\"></a>";
```

Editor window will open.

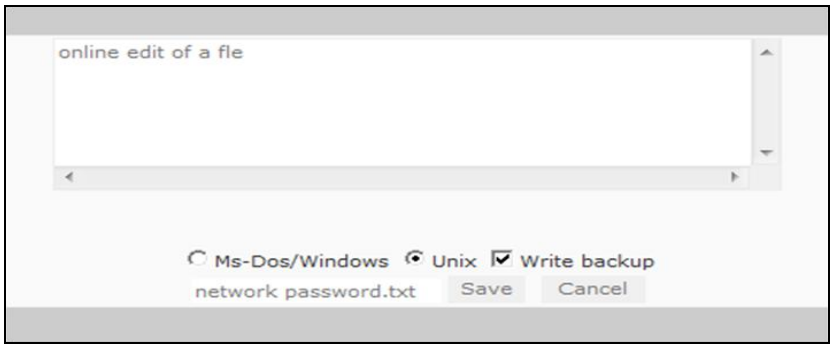



**Figure 5.19: Edit file.**

❖ **At server side:**

1. Open file.

```
//The number of colums for the edit field
int EDITFIELD_COLS = 100;
//The number of rows for the edit field
int EDITFIELD_ROWS = 50;
BufferedReader reader = new BufferedReader(new FileReader(ef));
String disable = "";
 if (!ef.canWrite()) disable = " readonly";
out.println("<form action=\""+browser_name+"\"method=\"Post\">\n"
   +"<textarea name=\"text\" wrap=\"off\" cols=\""+ EDITFIELD_COLS
   +"\" rows=\""+ EDITFIELD_ROWS + "\"" + disable + ">");
String c;
// Write out the file and check if it is a win or unix file
int i;
boolean dos = false;
boolean cr = false;
while ((i = reader.read()) >= 0) {
      out.print(conv2Html(i));
      if (i == '\r') cr = true;
      else if (cr && (i == '\n')) dos = true;
      else cr = false;}
reader.close();
```

2. Save change with backup.

```
File new_f = new File(getDir(f.getParent(),
request.getParameter("new_name")));
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
BufferedWriter outs = new BufferedWriter(new FileWriter(new_f));
StringReader text = new StringReader(request.getParameter("text"));
int i;
boolean cr = false;
String lineend = "\n";
if (request.getParameter("lineformat").equals("dos"))
        lineend = "\r\n";
while ((i = text.read()) >= 0) {
                if (i == '\r') cr = true;
                else if (i == '\n') {
                        outs.write(lineend);
                        cr = false;
                }else if (cr) {
                        outs.write(lineend);
                        cr = false;
                }else {
                        outs.write(i);
                        cr = false;}}
outs.flush();
outs.close();
```

3. Save backup.

```
File bak = new File(new_f.getAbsolutePath() + ".bak");
bak.delete();
new_f.renameTo(bak);
```

- **View file :**

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

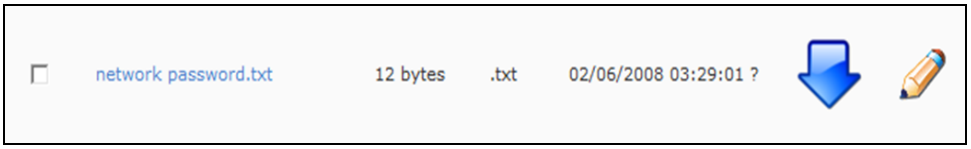



**Figure 5.20: view file.**

- **At client side:**

Files at shared folder view as a hyper link.

```
link= ahref+ "file=" + name + "\" target=\"_blank\">"+ buf + "</a>";
```

To view the file , user can choose download then choose open button. The file open with program install on client system . Or , double click on file , file open on browsers.

- **At server side:**

```
BufferedInputStream fileInput = new BufferedInputStream(new
FileInputStream(f));
byte buffer[] = new byte[8 * 1024];
out.clearBuffer();
OutputStream out_s = new Writer2Stream(out);
copyStreamsWithoutClose(fileInput, out_s, buffer);
fileInput.close();
out_s.flush();
```

### 5.2.5. Shared Board:

It is synchronous application that implemented by using the applet (discussed before) and the socket. We briefly take about the basics of TCPsocket for reliable communication as a base of our component implementation.

- **Sockets**

These classes are related to making normal TCP connections:

- ServerSocket
- Socket

For simple connections between a client and a server, **ServerSocket** and **Socket** are all that you will probably need.

**ServerSocket** represents the socket on a server that waits and listens for requests for service from a client.

**Socket** represents the endpoints for communication between a server and a client. When a server gets a request for service, it creates a Socket for communication with the client and continues to listen for other requests on the ServerSocket. The client also creates a Socket for communication with the server**.**

- **Reading / Writing Data**

Construct a DataInputStream or DataOutputStream object.

**A data input stream** lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

**A data output stream** lets an application write primitive Java data types to an output stream in a portable way. An application can then use a data input stream to read the data back in.

Both objects can be created using the socket's getInputStream() or getOutputStream() methods. Java allows to both read and write by applied some method as follow:

- **writeBoolean(boolean v):** writes a boolean to the underlying output stream as a 1-byte value.

- **writeInt(int v):**writes an int to the underlying output stream as four bytes, high byte first.

- **writeShort(int v):**writes a short to the underlying output stream as two bytes, high byte first.

- **writeUTF(String str):** writes a string to the underlying output stream using Java modified UTF-8 encoding in a machine-independent manner[32],[33].

**Shared Board Implementation:**

Firstly, to access to the shared board, click on its button in the group page that it opened the applet which embedded in **"applet.jsp"** contains the following:

```
<applet code="CFGPApplet.class" width="800" height="600" codebase="/applet">
    <param name="portnumber" value="5050">
    <param name="servername" value="localhost">
    <param name="password" value="1111">
    <param name="username" value=<%=session.getAttribute("name")%>>
    You 'll need a browser with Java support to run shared board
  </applet>
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

Here, it opens CFGPApplet that lied in applet directory and has the following parameters:

- username:

Using this flag followed by a user name automatically sets the user name in the client.

- password:

Using this flag followed by a password automatically sets the password in the client.

- servername:

This parameter allows you to set the server name in the client.

- portnumber

This option, followed by a numerical port value, to specify a TCP port as a default 12468.

CFGPApplet is an applet contains:

```
private static final long serialVersionUID = 220L;
```

We know that transport layer is only concerned with the transmission of sequences of bytes but if we have data structure as in our applet this layer does not distinguishes these data structure. So, we need to make the class serializable by adding this statement. Serializing an object involves encoding its state in a structured way within a byte array. Once an object is serialized, the byte array can be manipulated by sending over a network using a socket-based connection. This statement in included above most classes in program for serialization purpose.

In the init() function, we read all parameters from applet.jsp in applet tag:

```
public void init()
   {// Get the user name, password, host name, and port from the launching
        // document, if specified.
        if (getParameter(usernameParam) != null)
           name = getParameter(usernameParam);
```

```
if (getParameter(passwordParam) != null)
    password = getParameter(passwordParam);
if (getParameter(servernameParam) != null)
    host = getParameter(servernameParam);
if (getParameter(portnumberParam) != null)
    port = getParameter(portnumberParam);
if (getParameter(BoardroomParam) != null)
    room = getParameter(BoardroomParam);
// Done
return;}
```

In start() function, we make instance of CFGPPanel as a following:

```
// Launch a window.  It will show up as an applet window.
panel = new CFGPPanel(name, password, host, port, showCanvas);
```

Which its parameters are taken from the parameters of applet and showCanvas parameter is setting to true as a default for graphical display.

We supposed to attempt an automatic connection so, we make a little thread to do this separately from the applet initialization, since in many cases the applet will not be showing on the screen until after initialization, as a following:

```
(new Thread() {
        public void run()
            {
                // Wait until the panel is showing on the screen
                while (!panel.isVisible());
                // Go
                panel.offline();
                if (autoConnect)
                    panel.connect();
                // Is the user supposed to be placed in an initial
                // Board room?
                if (!room.equals("") && (panel.connection != null))
                    panel.connection.sendEnterRoom(room, false, "");
                    } }).start();
```

In the destroy() function, we close the connection of the server:

```
// Disconnect from the server if connected
        if (panel.connected == true)
            panel.disconnect();
```

In CFGPPanel, the drawing of graphical interface, we put some examples for creating its components:

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
// Set up all of the window stuff
        GridBagLayout layout = new GridBagLayout();
        setLayout(layout);
        // The top button panel
        Panel buttonPanel = new Panel();
        buttonPanel.setLayout(layout);
   //adding connect button
        connectButton = new Button("Connect");
        connectButton.setFont(largeBoldFont);
        connectButton.addActionListener(this);
        buttonPanel.add(connectButton,
                        new GridBagConstraints(0, 0, 1, 1, 1.0, 1.0,
                    GridBagConstraints.WEST,
                    GridBagConstraints.HORIZONTAL,
                     new Insets(0, 0, 0, 0), 0, 0));
                    GridBagConstraints.HORIZONTAL,
                    new Insets(0, 0, 0, 0), 0, 0));
```

A GridBagLayout places components in a grid of rows and columns, allowing specified components to span multiple rows or columns. Then, we create a panel that contains connect, disconnect, Boardrooms buttons.

Then, at the left of the center of the applet, we implement unedited area for presenting the events that occur in the applet, for example, others entering the room. As a following:

```
//area not editable
        messagesArea = new TextArea("", 7, 50,
                                    TextArea.SCROLLBARS_VERTICAL_ONLY);
        messagesArea.setEditable(false);
        messagesArea.setFont(smallFont);
        contentPanel.add(messagesArea,
                         new GridBagConstraints(0, 2, 2, 1, 1.0, 1.0,
                                                GridBagConstraints.NORTHWEST,
                                                GridBagConstraints.BOTH,
                                                new Insets(0, 0, 0, 0), 0, 0));
```

At the right of this area, we implement the control panel for represent to the user his activity (drawing) and the list of the users that they are online in the room:

```
//for presenting the user activity
        activityField = new TextField();
        activityField.setEditable(false);
        activityField.setFont(smallFont);
        controlsPanel
           .add(activityField,
                 new GridBagConstraints(0, 1, 1, 1, 1.0, 1.0,
                GridBagConstraints.NORTHWEST,
                                        GridBagConstraints.HORIZONTAL,
                                        new Insets(0, 0, 0, 0), 0, 0));
sendToList = new java.awt.List(4);
```

```
        sendToList.setFont(xSmallFont);
        sendToList.addItemListener(this);
        sendToList.setMultipleMode(true);
        controlsPanel.add(sendToList,
                        new GridBagConstraints(0, 3, 1, 1, 1.0, 1.0,
                                        GridBagConstraints.NORTHWEST,
                                        GridBagConstraints.BOTH,
                                        new Insets(0, 0, 0, 0),
                                        0, 0));
```

Then, at the bottom of the applet, in the left, the basic component of the applet, the shared board which is an instance of CFGPCanvas:

```
canvasPane = new ScrollPane(ScrollPane.SCROLLBARS_AS_NEEDED);
        canvas = new CFGPCanvas(this);
        canvasPane.add(canvas,
                        new GridBagConstraints(0, 0, 0, 1, 1.0, 1.0,
                                        GridBagConstraints.NORTHWEST,
                                        GridBagConstraints.BOTH,
                                        new Insets(0, 0, 0, 0),
                                        0, 0), 0);
```

This canvas has drawing tool that allowing the user to draw line or free hand, add text and select the color. Also, there is a button for clearCanvas for erase all graphics in the shared board.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

The basic functions of applet are the listening for key and mouse events. After listening, the action will be performed:

```
// register to receive the various events
        addKeyListener(this);
          addMouseListener(this);
```

The events that will be performed as a following:

- Connection by pressing connect button:

- The shared board will be clear.
- Initialize the message area.
- Open a client socket with passing the server name and the port as parameters.
- Make some buttons in the interface enable (disconnect, Board rooms buttons) and unable (connect button)

As a following:

```
        canvas.clear();
        synchronized (messagesArea)
           {messagesArea.setText("");   }
        labelDialog =
           new CFGPLabelDialog(parentWindow,
                                   "connect",
                                   "Connecting, one moment please...");


        // Open up my socket
```

```
            Socket socket = new Socket(host, portNumber);
            connection = new CFGPClientConnection(socket, this);
    // for disappearing the dialog
            labelDialog.dispose();
canvas.clear();
        synchronized (messagesArea)
            {messagesArea.setText("");  }
        labelDialog =
            new CFGPLabelDialog(parentWindow,
                                "connect",
                                "Connecting, one moment please...");
        // Open up my socket
            Socket socket = new Socket(host, portNumber);
            connection = new CFGPClientConnection(socket, this);
    // for disappearing the dialog
            labelDialog.dispose();
```

In the Constructor of CFGPClientConnection:

```
public CFGPClientConnection(Socket s, CFGPPanel panel)
        throws IOException
  {
        super(null, "CFGP Board client connection", s);
 // then Start listening for stuff from the server
        start();
```

In the constructor of parent (CFGPConnection which is thread), initialization of the streams that used by the socket at the two sides: server and client:

```
// Set up the streams
        iStream = new DataInputStream(socket.getInputStream());
        oStream = new DataOutputStream(socket.getOutputStream());
```

Then, in the client side, start listening to the server. This method starts the execution of the thread that lied in run method:

```
public void run()
  try {
                parseCommand();
// this method make the client listen to the command then perform the //appropriate action for it (Try to read a short from the stream.)
//          commandType = iStream.readShort();
//for example if the client read the command (ENTERROOM) so, that //means someone enter the room and the server tell you
}
```

Then, the client sends his information to the server:

```
synchronized (oStream)
                {
```

```
//to tell the server that the incoming user information
                    oStream.writeShort(CFGPCommand.USERINFO);
                    oStream.writeInt(id);
                    oStream.writeUTF(name);
                    oStream.writeUTF(password);}
```

At the Server side, the server must be executed by the command:

```
java CFGPServer –portnumber 5050
```

First, it open a server window for showing the users entered, leaved and a space for logging the events and shutdown button for disconnect the server. Then, open a server socket.

```
window = new CFGPServerWindow(this, "server");
        try {   serverSocket = new ServerSocket(port);}
```

Then, create a vector of connections and Boardrooms. Also, create a thread group that represents a set of threads. Then, create the initialization room ("Main"). Then, listen for the clients requests (which lied in run method that contains the same structure of client with different commands):

```
connections = new Vector();
        BoardRooms = new Vector();
        threadGroup = new ThreadGroup("CFGP Board server connections");

        // Create the initial 'main' Board room
```

```
mainRoom = new CFGPBoardRoom("Main", false, null);
start();
```

Because the client send its information, the server receives it:

```
// Receive information about a user
iStream.readInt(); // WE assign the user IDs
user.name = iStream.readUTF();
user.password = iStream.readUTF();
```

- Disconnection by pressing disconnect button:

- Close the connection
- Make some buttons in the interface enable (connect) and unable (disconnect, Board rooms buttons)

```
connection.stop = true;
        // Tell the server
connection.sendDisconnect();
connection.shutdown(false);
```

In the shutdown, close the streams, remove the userList in the interface and leave the room:

```
iStream.close();
        oStream.close();
// Empty out our user list
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
            userList.removeAllElements();
            // No more Board room
            mainPanel.currentRoom = null;
```

- Rooms list by pressing Boardrooms button:

It creates RoomDialog which has list of rooms:

```
roomsDialog = new CFGPRoomsDialog(this);
```

It has 4 buttons:

- enterRoom Button.
  The client select a preferred room then the system capture his selection then inform the server that he want to enter:

```
int index = 0;
CFGPRoomInfo selectedRoom;
index = roomList.getSelectedIndex();
selectedRoom = mainPanel.roomInfoArray[index];
if (selectedRoom.priv)
                    {
CFGPPasswordDialog passDialog =
        new CFGPPasswordDialog(mainPanel,
                                "enter password" +
                                " \"" + selectedRoom.name+ "\"", true);
                    password = passDialog.getPassword();
                }
```

The client sends to the server the following:

```
oStream.writeShort(CFGPCommand.ENTERROOM);
                    oStream.writeInt(fromId);
                    oStream.writeUTF(roomName);
                    oStream.writeBoolean(priv);//if required password
                    oStream.writeUTF(password);
```

Then, the server receives the following:

```
    iStream.readInt(); // user id
        newRoomName = iStream.readUTF();
        priv = iStream.readBoolean();
    password = iStream.readUTF();
// Create the Board room
newRoom = new CFGPBoardRoom(newRoomName, priv,password);
// Add it to the list of Board rooms
server.BoardRooms.addElement(newRoom);
```

Then, tell the connections about this updated:

```
synchronized (server.connections)
  {
        // Send the updated Board room list to all the clients
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
        for (int count = 0; count < server.currentConnections;
            count ++)
           {
tmpConnection = (CFGPServerConnection)
                    server.connections.elementAt(count);
                    tmpConnection.sendRoomList();
           }
   }
```

In the SendRoomList function the server tell the clients in detailed to tell each user in the rooms that this person enter the room:

```
ioStream.writeShort(CFGPCommand.ROOMLIST);
    oStream.writeShort(rooms.size());
        for (int count1 = 0; count1 < rooms.size();count1 ++)
        {room = (CFGPBoardRoom)rooms.elementAt(count1);
        oStream.writeUTF(room.name);
        oStream.writeBoolean(room.priv);
        oStream.writeInt(room.users.size());
        for (int count2 = 0;count2 < room.users.size(); count2 ++){
     CFGPServerConnection tmpConn =(CFGPServerConnection)
        room.users.elementAt(count2);
        oStream.writeUTF(tmpConn.user.name);}}
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

- Room Info Button.

The client selects the room from the list, then presses room info, and directly takes its properties from its data structure (CFGPRoomInfo):

```
selectedRoom = mainPanel.roomInfoArray[index];
infoString += "room name" +
   " " + selectedRoom.name + "\n" +
  "private"+ " ";
if (selectedRoom.priv)//require password
   {infoString += "yes" + "\n" +    " "; }
else
 infoString += "no";
infoString += "\n" +
   "users" + " " +
   selectedRoom.userNames.size() + "\n\n";
```

- Create Room Button.

If the client wants to create press on this button then a new dialog is presented and asking for room name and password if he want to make the room private:

```
new CreateRoomDialog(mainPanel);
```

Then, the system creates the new room and makes the client entering this room so, use the same previous method to enter room:

```
mainPanel.canvas.clear();
  mainPanel.connection.sendEnterRoom(roomNameField.getText(),
 privCheckbox.getState(), pword);
```

```
mainPanel.currentRoom.name = roomNameField.getText();
```

- Update List Button

Here, the client asks the server for updating so, send to the server the request:

```
mainPanel.connection.sendRoomList(null);
```

In the send room list just send 0 as a short number:

```
            // Asking for a list
                        oStream.writeShort(0);
```

At the server side: it read the null number so, the server knows that the client wants to update list. Then, it called sendRoomList as illustrated before.

```
// The client is requesting a list of the available Board rooms.
        // Read the null 'number' value from the stream
        iStream.readShort();
        // Now we send the list
                sendRoomList();
```

**- Showing Canvas:**

Here, it is an instance of Canvas. The main idea is rotated on the listener for mouse events:

- **mousePressed(MouseEvent E):** Invoked when a mouse button has been pressed on a component:

```
// Save the coordinates of the initial click
        startX = E.getX();
        startY = E.getY();
        // If we are doing lines, we will show
        // draft lines to suggest the final shape of the object
        if (drawType == CFGPCanvas.LINE)
           {draftLine = new Line(drawColor, startX, startY, startX, startY);
```

In the constructor of the line, we determine the color of line and the coordinates of the line as a point firstly. Then set the draw mode to XOR (its value 1) and draw it

```
drawLine(draftLine.color, draftLine.startX,
                draftLine.startY, draftLine.endX,
                draftLine.endY,
                CFGPCanvas.MODE_XOR);
```

We apply XOR mode in drawLine function, by setting the paint mode of this graphics context to alternate between graphics context's current color (white color "ffffff" hex code) and the new specified color.

- **mouseReleased(MouseEvent E):** Invoked when a mouse button has been released on a component.

  It invoked when the client press on text checkbox, then press the mouse and released it, then set the draw mode to MODE_XOR (its value 1):

```
drawText(drawColor, draftText.startX, draftText.startY,
                                draftText.fontNumber, draftText.attribs,
```

```
draftText.size, draftText.text,

CFGPCanvas. MODE_XOR);
```

Then, it will be send as writing its parameters with ostream to be received from the server that reading it by istream then sending to all members in the group by writing with ostream, then save this line with its information and the drawer to be permanent:

```
// Add this text to our canvas vector

        user.Boardroom.canvasHistory.addElement(

            new CanvasCommand(CFGPCommand.DRAWTEXT, user.id,
colour, startx, starty, endx, endy, (short) 0, (short) 0,(short) 0, null));

//the last 4 parameters   type ,attributes ,size , text
```

- **mouseDragged(MouseEvent E):** Invoked when a mouse button is pressed on a component and then dragged.

The line in the mouse pressed is just a draft line because the end of the line is not specified. After dragging the end of the line occurs and it will be permanent:

```
draftLine.endX = E.getX();

draftLine.endY = E.getY();

drawLine(draftLine.color, draftLine.startX, draftLine.startY,
draftLine.endX,draftLine.endY,CFGPCanvas.MODE_XOR);
```

Then, it will be sending as a text above then be saved:

```
// Add this line to our canvas vector

        user.Boardroom.canvasHistory.addElement(
```

```
                    new CanvasCommand(CFGPCommand.LINE, user.id, colour,
startx, starty, endx, endy, (short) 0, (short) 0,(short) 0, null));

//the last 4 parameters   type , attributes ,size , text
```

### 5.2.6. Log file:

- **At server side**

With any event e.g. add task, insert this event in **log file** table,as following.

```
String contant= name+ ":  Add Task which is [ "+ task +" In ( "+day+" / "+month+" /
"+yeer+" ) ,Time is: "+time;

query =  "INSERT INTO logfile(content,group_name,action_date)values(?,?, now())";

 PreparedStatement stmtt = db.Prequery(query);

                    stmtt.setString( 1, contant);

                    stmtt.setString( 2, group);

                    // Send query to database.

                   resultSet= stmtt.executeUpdate();
```

Log file implement by select the ten largest logf_no (last ten event) from table **log file** by while loop, as following.

```
ResultSet rs = db.selectQuery("SELECT MAX(logf_no) logf_no FROM logfile WHERE
group_name =\"" + sessionGroup + "\"");

int lognum=0;

while(rs.next()){

   lognum = rs.getInt("logf_no");}

while (count<10){

   rs1 = db.selectQuery("SELECT *  FROM logfile where logf_no =\""+ lognum +"\" and
group_name=\"" + sessionGroup + "\"");
```

```
    while(rs1.next()) {
        content = rs1.getString("content");
        date = rs1.getString("action_date");
        count=count+1; }
   lognum=lognum-1;
 } //while
```

Also, delete others event that we do not need it, as following.

```
while (lognum>=0){
   String query = "DELETE FROM logfile WHERE logf_no =\""+ lognum   +"\"and
group_name=\"" + sessionGroup + "\"" ;
    PreparedStatement stmtt = db.Prequery(query);
      int  resultSet = stmtt.executeUpdate();
                    lognum=lognum-1;}
```

- **At client side:**

Member can view a list last ten events as (figure 5.21).

| Event | Date |
|---|---|
| may: Add Rerlay to This Advertisment(click here) | 2008-05-02 05:00:12 |
| zainab: Add Rerlay to This Advertisment(click here) | 2008-05-02 04:59:15 |
| reem: Add Rerlay to This Advertisment(click here) | 2008-05-02 04:52:46 |
| sara: Add Rerlay to This Advertisment(click here) | 2008-05-02 04:50:45 |
| basemah: Add vote which is( Is [ 1/5/1429 ] in [ 5 pm ] sutable for you to holding a meeting ? | 2008-05-02 04:35:25 |
| basemah: Add Advertisment to This Section (click here) | 2008-05-02 04:27:04 |
| basemah: Edit Task to: [ write source code In ( 9 / June / 2008 ) ,Time is: 1:50 | 2008-05-02 04:19:37 |
| basemah: Delete Task | 2008-05-02 04:14:37 |
| basemah: Add Task which is [ please comlete the report In ( 12 / June / 2008 ) ,Time is: 10 am | 2008-05-02 04:07:32 |
| maha: Add Rerlay to This Advertisment(click here) | 2008-05-02 03:51:40 |

**Figure 5.21: last 10 event.**

### 5.2.7 Advertising space:

Advertising Space contains many sections, these sections include many advertisements. The last one can have number of replays .

- **Add Sections**

❖ **At client side:**

Create sections assumed to be processed by group's coordinator. From browser, coordinator fills the Add section form for required data see figure (5.22). These data submitted as a request to add section servlet handler at server side to process them.

**Figure 5.22: add section.**

❖ **At server side:**

Add section handler servlet gets Data as requests from an add section form at client side. Then, connecting to CFGP database and performing queries .If no error occurs, add section completed by accessing table **Forum_forums** of CFGP database. As following:

```
String query = "INSERT INTO forum_forums
(forum_id,title,forum_info,Group_name)values(?,?,?,?) " ;
PreparedStatement stmtt = db.Prequery(query);
                                    stmtt.setString( 1, forum_id);
                                    stmtt.setString( 2, title);
                                    stmtt.setString( 3, forum_info);
                                   stmtt.setString( 4, sessionGroup);
                          // Send query to database.
```

```
int resultSet = stmtt.executeUpdate();
```

- **View Sections**

❖ **At server side:**

To view all sections, a query sends to table **forum_forums** of CFGP database. As following:

```
ResultSet rs = db.selectQuery("SELECT * FROM forum_forums WHERE group_name =\""
+ sessionGroup + "\" ORDER BY title");
```

❖ **At client side:**

Sections are displayed as figure (5.23).

Advertising Space

| Section | Topics | Replies | Last Post Info | Delete Section |
|---|---|---|---|---|
| Achievements | 1 | 0 | 2008-05-01 10:44:08.0<br>In: voting<br>By: basemah | Delete |
| Meeting<br>with Supervisor. | 4 | 0 | 2008-05-01 22:16:19.0<br>In: Meeting in 15/5/1429<br>By: System | Delete |
| ToDo | 2 | 1 | 2008-04-30 07:38:17.0<br>In: to complete#2<br>By: basemah | Delete |

**Figure 5.23: advertising space**

- **Delete Section**

❖ **At client side:**

Delete sections assumed to be processed by group's coordinator. From browser, coordinator press delete section link see figure (5.24). This process submitted as a request to delete section servlet handler at server side to process them.

Advertising Space

| Section | Topics | Replies | Last Post Info | Delete Section |
|---|---|---|---|---|
| Achievements | 1 | 0 | 2008-05-01 10:44:08.0<br>In: voting<br>By: basemah | Delete |

**Figure 5.24: delete section**

❖ **At server side:**

Delete section handler servlet gets Data as requests from delete section link at client side. Then, connecting to CFGP database and performing queries .If no error occurs, delete section completed by accessing appropriate tables of CFGP database.

To delete a section, a query send to table **forum_message** to delete all replays in this section, then send to table **forum_threads** to delete all advertisements in this section , finally send to table **forum_forums** to delete this section. As following:

```
String query = "DELETE FROM forum_message WHERE forum_id=\"" + forum_id + "\"" ;
 String query2 = "DELETE FROM forum_threads WHERE forum_id=\"" + forum_id + "\"" ;
String query3 = "DELETE FROM forum_forums WHERE forum_id=\"" + forum_id + "\"" ;
  PreparedStatement stmtt = db.Prequery(query);
  int  resultSet = stmtt.executeUpdate();
  stmtt = db.Prequery(query2);
  resultSet = stmtt.executeUpdate();
  stmtt = db.Prequery(query3);
   resultSet = stmtt.executeUpdate();
```

▪ **Add Advertisement:**

❖ **At client side:**

Add Advertisement allowed to be processed by all group's member. From browser, member fills the Add Advertisement form for required data see figure (5.25). These data submitted as a request to add advertisement servlet handler at server side to process them.

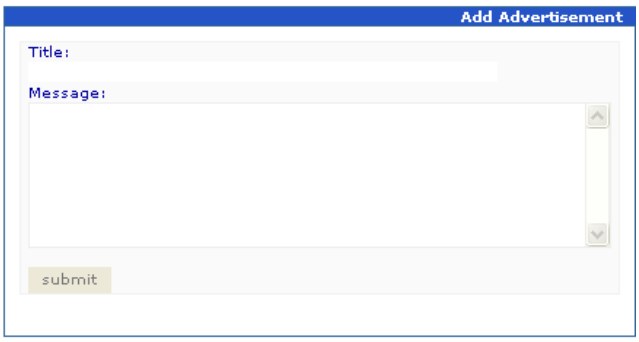



**Figure 5.25: add advertisement.**

- **At server side:**

Add Advertisement handler servlet gets Data as requests from an add Advertisement form at client side. Then, connecting to CFGP database and performing queries .If no error occurs add Advertisement completed by accessing appropriate tables of CFGP database.

To add Advertisement, a query sends to table **forum_thread** to insert Advertisement title. As following:

```
String query ="INSERT INTO forum_threads
(forum_id,thread_id,Group_name,title,views)values(?,?,?,?,?) " ;
PreparedStatement stmtt = db.Prequery(query);
                    stmtt.setString( 1, forum_id);
                    stmtt.setString( 2, thread_id);
                stmtt.setString( 3,sessionGroup);
                    stmtt.setString( 4, title);
                    stmtt.setString( 5,"0");
                    // Send query to database.
                    int  resultSet = stmtt.executeUpdate();
```

Then send aquery to table **forum_message** to insert advertisement message. As following:

```
query =  "INSERT INTO forum_message
(forum_id,thread_id,reply_id,Group_name,message,user,user_type,date_time)values(?,?,?,?,?,?,?,?,now()) " ;
 stmtt = db.Prequery(query);
                    stmtt.setString( 1, forum_id);
                    stmtt.setString( 2, thread_id);
                    stmtt.setString( 3, "0");
                    stmtt.setString( 4, sessionGroup);
                    stmtt.setString( 5, message);
                    stmtt.setString( 6, user);
                    stmtt.setString( 7, sessionType);
                    // Send query to database.
                    resultSet = stmtt.executeUpdate();
```

- **View advertisements**

❖ **At server side:**

To view all advertisements, a query sends to table **forum_message** of CFGP database. As following:

```
ResultSet rs2 = db.selectQuery("SELECT * "+"FROM forum_message "+
"WHERE forum_id=\"" + forum_id + "\" " + "AND thread_id=\""+ thread_id + "\" " +"AND reply_id =\"0\"");
```

❖ **At client side:**

Advertisements are displayed as figure (5.26).

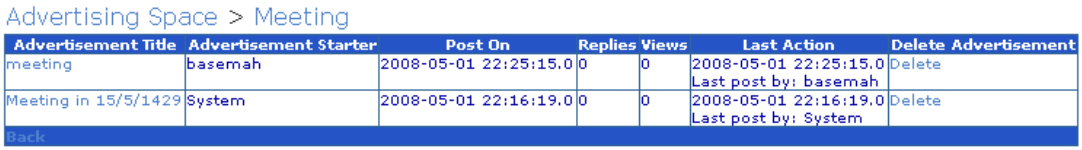



**Figure 5.26: advertisements.**

- **Delete advertisements**

❖ **At client side:**

Delete advertisements assumed to be processed by group's coordinator. From browser, coordinator press delete advertisement link see figure (5.27). This process submitted as a request to delete advertisements servlet handler at server side to process them.

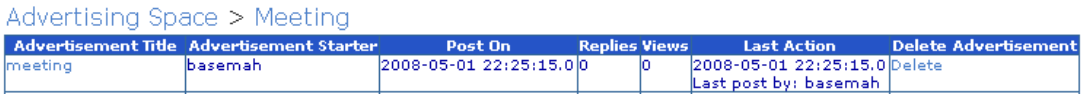



**Figure 5.27: delete advertisement.**

❖ **At server side:**

Delete advertisements handler servlet gets Data as requests from delete advertisements link at client side. Then, connecting to CFGP database and performing queries .If no error occurs, delete advertisement completed by accessing appropriate tables of CFGP database.

To delete an advertisement, a query send to table **forum_message** to delete all replays in this advertisement, then send to table **forum_threads** to delete advertisement. As following:

```
String query = "DELETE FROM forum_message WHERE forum_id=\"" + forum_id + "\"
AND thread_id=\"" + thread_id + "\"" ;

String query2 = "DELETE FROM forum_threads WHERE forum_id=\"" + forum_id + "\"
AND thread_id=\"" + thread_id + "\"";

        PreparedStatement stmtt = db.Prequery(query);
```

```
int  resultSet = stmtt.executeUpdate();
stmtt = db.Prequery(query2);
// Send query to database.
resultSet = stmtt.executeUpdate();
```

- **Add replay:**

❖ **At client side:**

Add replay allowed to be processed by all group's member. From browser, member fills the Add replay form for required data see figure (5.28). These data submitted as a request to add replay servlet handler at server side to process them.

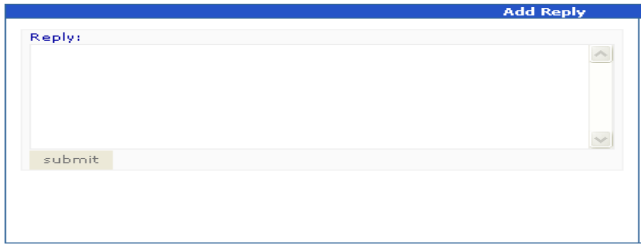



**Figure 5.28: add reply.**

❖ **At server side:**

Add replay handler servlet gets Data as requests from an add replay form at client side. Then, connecting to CFGP database and performing queries .If no error occurs add replay completed by accessing table **Forum_message** in CFGP database. As following:

```
String query = "INSERT INTO forum_message
(forum_id,thread_id,reply_id,Group_name,message,user,user_type,date_time)values(?,?,?,?,?,?,?,?,now()) " ;
PreparedStatement stmtt = db.Prequery(query);
```

```
                    stmtt.setString( 1, forum_id);
                    stmtt.setString( 2, thread_id);
                    stmtt.setString( 3, thread_id );
                    stmtt.setString( 4, sessionGroup);
                    stmtt.setString( 5, message);
                    stmtt.setString( 6, user);
                    stmtt.setString( 7, sessionType);
                    // Send query to database.
            int  resultSet = stmtt.executeUpdate();
```

- **View replays**

❖ **At server side:**

To view all replays in specific advertisement, a query send to table **forum_message** of CFGP database. As following:

```
ResultSet rs = db.selectQuery(
                "SELECT * FROM forum_message " +
                "WHERE forum_id=\"" + forum_id + "\" AND thread_id =\"" +
reqThread_id + "\" AND group_name =\"" + sessionGroup + "\" ORDER BY reply_id LIMIT
"+ start + "," + Variable.getMessagePerPage());


                while(rs.next()){
                    thread_id = rs.getString("thread_id");
                    String reply_id = rs.getString("reply_id");
                    String message = rs.getString("message");
                    String user = rs.getString("user");
                    String date_time = rs.getString("date_time");
                    String groupname = rs.getString("group_name");
```

```
String userType = rs.getString("user_type");
```

❖ **At client side:**

Replays are displayed as figure (5.29).

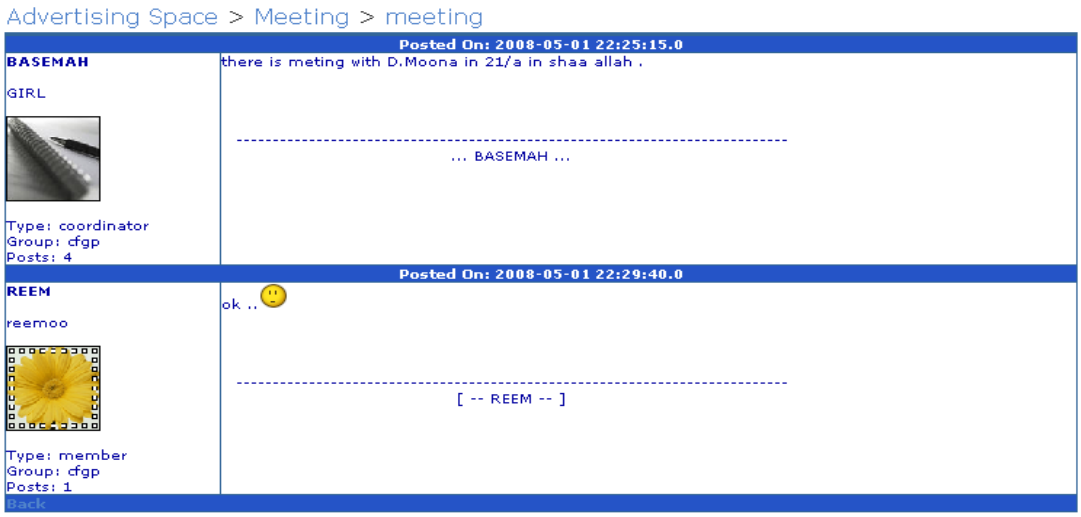



**Figure 5.29: view an advertisement.**

▪ **Delete replays**

❖ **At client side**:

Delete replay assumed to be processed by group's coordinator. From browser, coordinator press delete replay link see figure (5.30). This process submitted as a request to delete replay servlet handler at server side to process them.

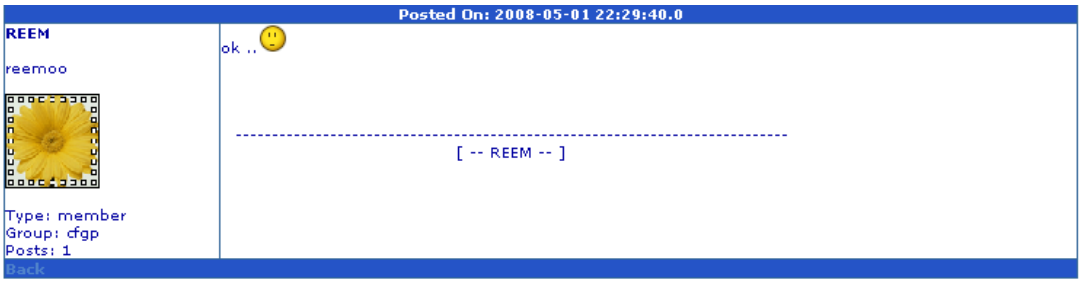



**Figure 5.30: delete reply.**

❖ **At server side:**

Delete replay handler servlet gets Data as requests from delete replay link at client side. Then, connecting to CFGP database and performing queries .If no error occurs, delete replay completed by accessing table **forum_message** in CFGP database. As following:

```
String query = "DELETE FROM forum_message WHERE forum_id=\"" + forum_id + "\"
AND thread_id=\""+ thread_id + "\" AND reply_id=\"" + reply_id + "\"";

        PreparedStatement stmtt = db.Prequery(query);

        // Send query to database.

        int  resultSet = stmtt.executeUpdate();
```

## 5.2.8 Voting module:

- **Create new vote**

❖ **At client side:**

When a supervisor or a coordinator desire to add a vote they need to fill a form to determine the time and the date for the meeting.

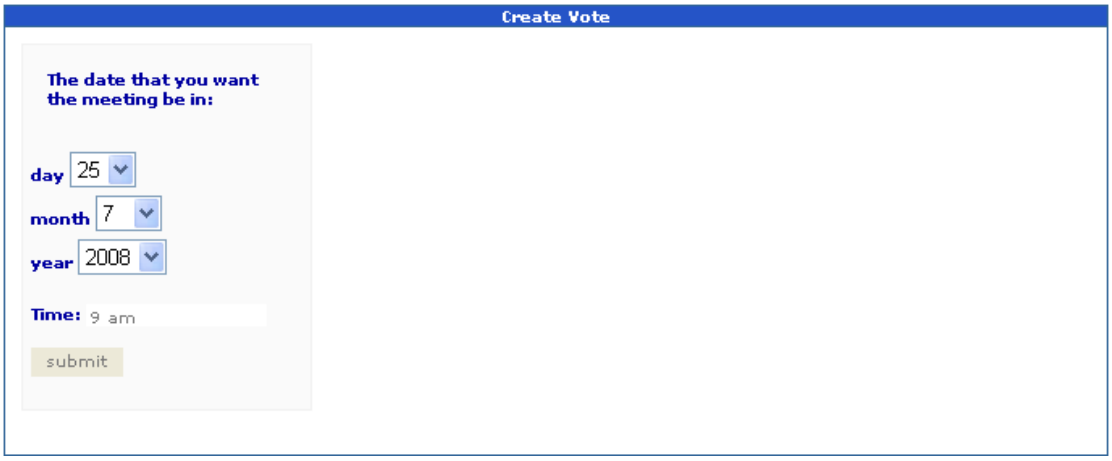



**Figure 5.31: create vote.**

- **At server side:**

After pressing submit button by the client the sever get these information.

```
String day = request.getParameter("day");
String month = request.getParameter("month");
String year = request.getParameter("year");
String time = request.getParameter("time");
```

The username and his/her type with a group name come by session form the previous page.

This information should be inserts to the vote table in data base

```
String query = "INSERT INTO votev
(name,group_name,vdate,vmonth,vyear,vtime,vote1,vote2)values(?,?,?,?,?,?,?,?) " ;
        PreparedStatement stmtt = db.Prequery(query);
                stmtt.setString(1,sessionUsername);
                stmtt.setString(2,sessionGroup );
                stmtt.setString(3,day);
                stmtt.setString(4,month);
                stmtt.setString(5,year);
                stmtt.setString(6,time);
```

```
                                        stmtt.setInt(7,0);

                                        stmtt.setInt(8,0);

int  resultSet = stmtt.executeUpdate();String query

                stmtt.setInt(6,0);

                int  resultSet = stmtt.executeUpdate();
```

Also, insert it to the log table

```
String contant= sessionUsername+ ":  Add vote which is( Is [ "+day+"/"+month+"/"+year+" ] in [ "+time+" ] sutable for you to holding a meeting ? ";


query =  "INSERT INTO logfile (content,group_name,action_date)values(?,?, now()) " ;

 stmtt = db.Prequery(query);

                                        stmtt.setString( 1, contant);

                                        stmtt.setString( 2, sessionGroup);

                                        // Send query to database.

                                        resultSet = stmtt.executeUpdate();
```

- **View Vote**

  - **At client side:**

When a user open the site a meeting vote form or a result of voting of that meeting will automatically display in the CFGP web-based tool

  - **At server side:**

The server selects the last vote done by supervisor or coordinator.

```
ResultSet rs = db.selectQuery("SELECT MAX(vote_no) vote_no FROM votev WHERE group_name =\"" + sessionGroup + "\"");

                        int  v_no = 0;
```

```
                while(rs.next()){
                    v_no = rs.getInt("vote_no");  }
 String query= " SELECT * FROM  votev  where vote_no='"+v_no+"' and group_name =\""
+ sessionGroup + "\"";
            ResultSet rs2 = db.selectQuery(query);
                                String name = "null";
                                String vdate = "null";
                                String vmonth = "null";
                                String vyear = "null";
                                String vtime = "null";
                                int v1=0;
                                int v2=0;
                        while(rs2.next()){
                          v1= rs2.getInt("vote1");
                          v2= rs2.getInt("vote2");
                      name = rs2.getString("name");
                      vdate = rs2.getString("vdate");
                          vmonth = rs2.getString("vmonth");
                          vyear = rs2.getString("vyear");
                          vtime = rs2.getString("vtime");}
```

Then, will parse the determined date " day , month and year "from string  to integer and extract from the day and month one as following

```
int vday = Integer.parseInt(vdate);
int imonth = Integer.parseInt(vmonth);
                        int iyear = Integer.parseInt(vyear);
                        int iday = vday - 1;
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
int mmonth = imonth - 1;
```

Using " **import** ='java.util.Calendar'" the program will extract the day , month and year of the computer as following.

```
Calendar cal = Calendar.getInstance();
int day = cal.get(Calendar.DATE);
int month = cal.get(Calendar.MONTH) + 1;
int year = cal.get(Calendar.YEAR);
```

If the day or month of meeting is 1 so the JSP will let day before 24 hour of meeting is 29 and month is 12 as following.

```
if (iday == 0){ mday = 29; }%>

if (mmonth == 0){ mmonth = 12; }%>
```

If the day of the system is before 24 hour of the meeting date suggested by coordinator or supervisor. The JSP page will display a form if there is a meeting or not according to the number of "yes" choice and "No" choice as following.

```
if (( day == iday)&& (month == imonth)&& ( year == iyear) )

<form name="dateVote" class="login" method="post">

        <p1 ALIGN="center"><h1>|| Meeting date || </h1></p1>

        <% if ( v1 > v2 ) { %>

<p>There is a meeting in [ <%= vdate%> / <%= vmonth%> / <%= vyear%>]  at [ <%= vtime%> ] please be therae at time  </p>

        <p1 ALIGN="center"><h1>Thank you !  </h1></p1>

        <% }else{ %>

<p> There is no meeting in [ <%= vdate%> / <%= vmonth%> / <%= vyear%>]     at [ <%=vtime%> ] please wait for another notification  </p>

        <% } %>
```

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

```
else if (( day == mday )&& (month <= mmonth)&& ( year <= iyear))

<form name="dateVote" class="login" method="post">

        <p1 ALIGN="center"><h1>|| Meeting date || </h1></p1>

        <% if ( v1 > v2 ) { %>

<p>There is a meeting in [ <%= vdate%> / <%= vmonth%> / <%= vyear%>]  at [ <%= vtime%> ] please be therae at time  </p>

        <p1 ALIGN="center"><h1>Thank you !  </h1></p1>

        <% }else{ %>

<p> There is no meeting in [ <%= vdate%> / <%= vmonth%> / <%= vyear%>]     at [ <%=vtime%> ] please wait for another notification  </p>

        <% } %> }
```

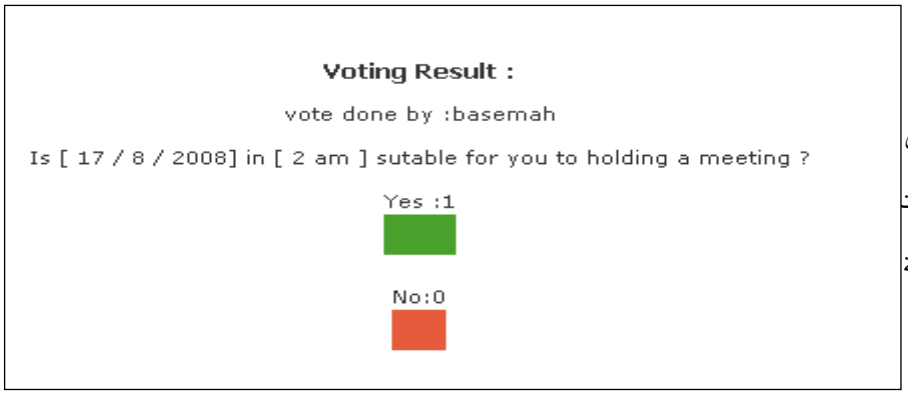
But if day of the system is less than that date the last voting form will display which is in the memberv.jsp file as following

```
 else if(( day < iday) && (month == imonth) && ( year == iyear) ){ %>

        <jsp:include page="memberv.jsp" flush="true" />

else if((month < imonth)  && ( year <= iyear)){%>

        <jsp:include page="memberv.jsp" flush="true" />

        <% } %>
```

- **Voting result**

  - **At client side**

When a user vote a result of voting will display




مها الفوزان

قسم مهارات

عمادة السنة

**Figure 5.32: voting result.**

- **At server side:**

First of all a servlet will get a vote number from voting form after a member press submit and connect to the database as following.

```
String no = request.getParameter("qid");
                        db.connect();
```

Then, select the information of the vote that has this vote number and get the choice of the user if yes or no as following.

```
ResultSet rs = db.selectQuery("SELECT * FROM votev WHERE vote_no=\"" + no + "\"");
                        while(rs.next()){
                         v1= rs.getInt("vote1");
                        v2= rs.getInt("vote2");}
   String opinion = request.getParameter("opinion")
```

What ever the choice is the servlet add to the choice one and update the database then display the result as shown below.

```
        if ( opinion.equals("yes"))
{String query ="UPDATE votev SET vote1 =? WHERE vote_no =\"" + no + "\"";
                        v1 = v1+1;
   PreparedStatement stmtt = db.Prequery(query);
   stmtt.setInt(1,v1);
   int resultSet=stmtt.executeUpdate();
}else
{String query ="UPDATE votev SET vote2 =? WHERE vote_no =\"" + no + "\"";
```

```
                    v2 = v2+1;


            PreparedStatement stmtt = db.Prequery(query);
            stmtt.setInt(1,v2);
            int resultSet=stmtt.executeUpdate();
```

Then the result of voting will displayed as following .

```
<p1 ALIGN="left"><h1> Voting Result :<%= ugname%></h1></p1>
 vote done by :<%= name%>
<p>Is [ <%= vdate%> ] in [ <%= vtime%> ] sutable for you to holding a meeting ? </p>
 <div id="content-wrap">
<TABLE cellSpacing="0" cellPadding="0" width=<%= v1%>   border="0" align="center">
Yes :<%= v1%> <TR bgColor="#4AA02C" height="25px">
<TD align="left" valign="top"/>
</TR>
</TABLE>
```

### 5.2.9. Task management:

Task schedule page shows the task schedule with date and time and allows coordinator to add, edit and delete task.

- **View Task schedule**
  - **At server side:**

View task schedule is implementing by send a query to database. This query selects all records in task table, as following.

```
ResultSet rs = db.selectQuery("SELECT * FROM task WHERE name =\"" + name + "\"");
```

❖ **At client side:**

Task schedule is displayed as figure (5.33).

| Task | Day | Month | Year | Time | Edit | Delete |
|---|---|---|---|---|---|---|
| planing in paper | 10 | May | 2008 | 10:30 | | |
| write chapter introduction in report | 11 | May | 2008 | 5 pm | | |
| write source code | 9 | June | 2008 | 1:50 | | |
| please comlete the report | 12 | June | 2008 | 10 am | | |

**Figure 5.33: Task schedule.**

- **Add Task**

❖ **At client side:**

If the coordinator wants add task, he/she should click "add" figure( 5.34) then the server open page contain form of task information figure( 5.35) , After coordinator fill the form he/she should click ok.

**Figure 5.34:add task button.**

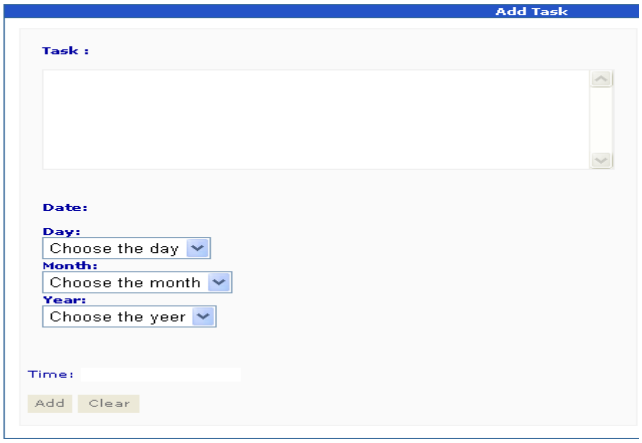



**Figure 5.35:add task.**

- **At server side**:

The query sends to database to add the task to table **task,** as following**.**

```
PreparedStatement st;
 String query="INSERT INTO task (name,task,day,month,yeer,time) VALUES (?,?,?,?,?,?)";
st = db.Prequery(query);
            st.setString(1,name);
            st.setString(2,task);
            st.setString(3,day);
            st.setString(4,month);
            st.setString(5,yeer);
            st.setString(6,time);
       int  resultSet = st.executeUpdate();
```

- **Edit Task**

❖ **At client side:**

If the coordinator wants edit task he/she should click "edit"(see figure 5.33) then the server open page which contain information about the task(see figure 5.36) that he/she wants edit it, after coordinator edits task, he/she click "Edit".

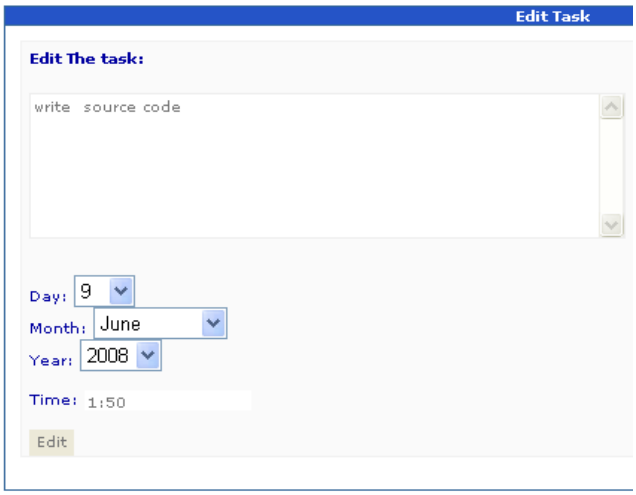



**Figure 5.36: Edit task.**

❖ **At server side:**

The query sends to database to edit the task from table **task,** as following .

```
PreparedStatement pstmt;

int numUpd ;

String query="UPDATE task SET task=?, day = ?, month=?, yeer=? ,time=?  WHERE
task_no = '"+ row +"'";

pstmt = db.Prequery(query);

pstmt.setString(1,task);        // Assign value to first parameter

pstmt.setString(2,day);       // Assign value to second parameter

pstmt.setString(3,month);

pstmt.setString(4,yeer);

pstmt.setString(5,time);

    numUpd =pstmt.executeUpdate();   // Perform the update
```

- **Delete Task**

❖ **At client side:**

If the coordinator wants delete task he/she click "delete"(see figure 5.33).

❖ **At server side:**

The query sends to database to delete the task from table **task,** as following.

```
String query = "DELETE FROM task WHERE task_no=\"" + row + "\
PreparedStatement stmtt = db.Prequery(query);

                    int  resultSet = stmtt.executeUpdate();
```

## 5.2.10. Contact List:

When student's login success, student will be add to **list** table in CFGP data base. And when student Log out, his/her record will be deleted from **List** table.

❖ **At server side**

Contact List is implementing by send a query to table **list** in CFGP database. This query selects all records which belong to student's group. As following:

```
ResultSet rs = db.selectQuery("SELECT * FROM list WHERE

 group_name =\"" + sessionGroup + "\" ORDER BY name");

while(rs.next()){

        String name = rs.getString("name");

        String type = rs.getString("type");
```

❖ **At client side:**

Member can view a list of all (on-line) members as figure (5.37).

**Figure 5.37: contact list.**

## 5.3 Conclusion

In conclusion, this chapter has provided a details of implementation of CFGP web based tool. It gave a closer look by demonstrating the system in a graphical/textual way from the logging in to the logging off procedure. In addition, this chapter provides the reasons of selecting programming language and database system.

# 6. Test

After complete the implementation, it was necessary to test CFGP web based tool. The main goal of testing is to allow the client to browse CFGP web based tool easily and to use all the components which helps to effect the web with other members of a group.

We connect five labtops , four of them are clients and one is a server by wierless connection. Each of them is with an IP address as below :

Server : 192.168.1.1

Client1 : 192.168.1.2

Client2 : 192.168.1.3

Client3 : 192.168.1.4

Client4 : 192.168.1.5

Then we test the following features of the CFGP web based tool.

## 6.1. Log in

The client needs to log in with his/her user name and password in addition to his/her group name as shown in the figure (6.1).

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

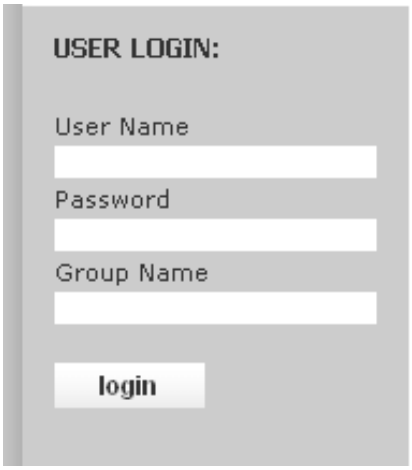



**Figure 6. 1: login form**

## 6.2. Registration

If the client belongs to an existent group, he/she must register as a new member of a group. However he can register a new group as group's coordinator as shown in figure (6.2).

**Figure 6. 2: Registration**

### 6.2.1. Register as a member of a group

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

In order to be a member of group, team member must be authenticated by group's name and password, as following figure (6.3).

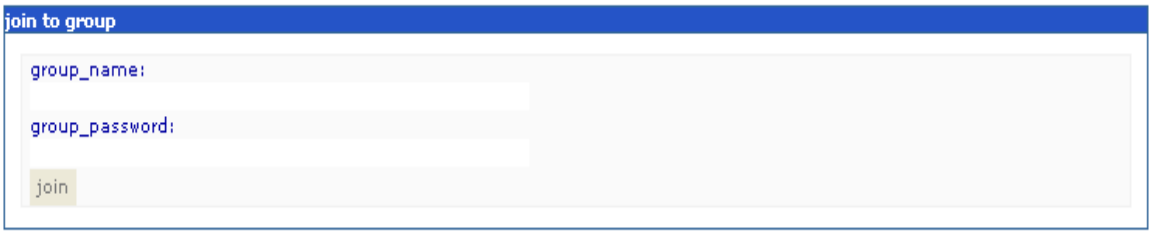



**Figure 6. 3: join to group.**

An explanation will be shown if group's data was invalid, as shown in figure (6.4).

**Figure 6. 4:wrong group name.**

If group name and password was correct the registration form will appear as a following:

**Figure 6. 5: new member of an existed group**

Notification message will display if register's name existed in the same group as shown in figure (6.6).

**Figure 6. 6: username exists in group**

The client must enter a valid email or a warning box will appear as a following.

**Figure 6. 7: invalid Email**

Figure (6.8) displays if a client try to choose a supervisor choice while there is another member registered as supervisor of a group. Each group must have only one supervisor.

**Figure 6. 8: a supervisor existed in a group.**

A message box will appear when registration is complete as following.

**Figure 6. 9: regestration successfuly completed**

### 6.2.2 Register as a new group:

The person who will create group's work space is assumed to be a coordinator of a group and he/she must enter the following information as shown in figure (6.10).

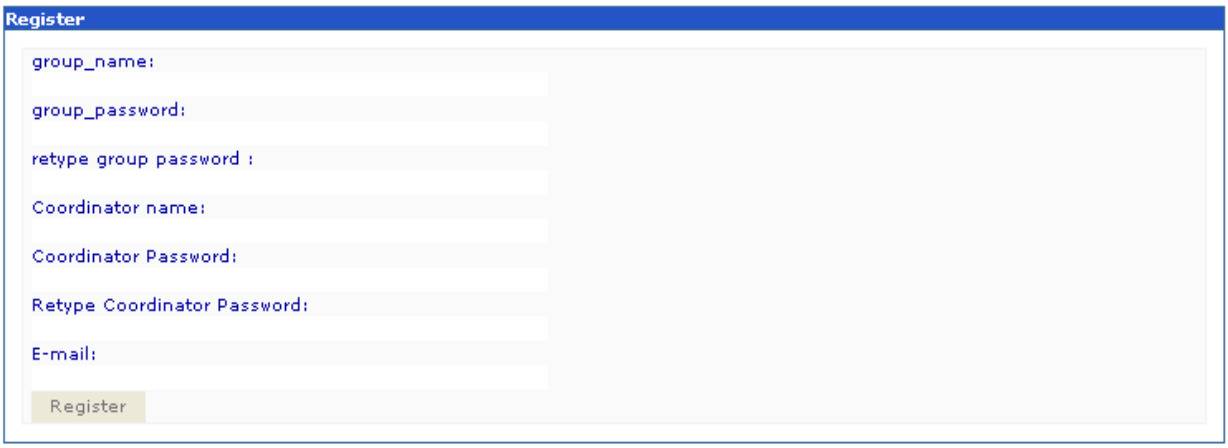



**Figure 6. :10 new group registration.**

After log in, a member name, member type and group name will be displayed in the group work page as following:

**Figure 6. 11: member information.**

And for each member will display appropriate page according to member type as following:

- **supervisor page**

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

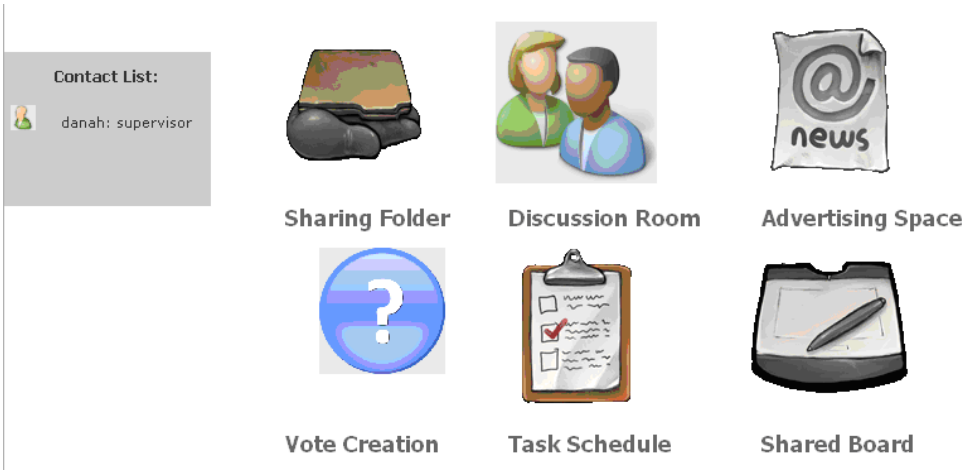



**Figure 6. 12:supervisor page.**

## ❖ A coordinator page

**Figure 6. 13:coordinator page.**

## ❖ A member page

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

**Figure 6. 14: member page.**

## 6.3 Contact List

A user can see his/her group members list if they are connect the CFGP web based tool or not as following:

**Figure 6. 15:contact list.**

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

## 6.4 Shared Board

The shared board is an applet executed by Java Virtual Machine (JVM) see figure (6.16).

**Figure 6. 16:Java**

When press a shared icon figure (6.17).

**Figure 6. 17:shared board icon.**

It will connect to the server as shown in figure (6.18).

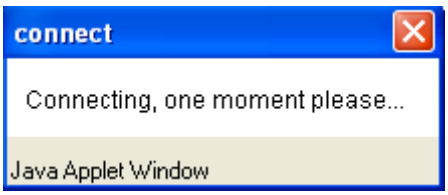



**Figure 6. 18:connecting.**

The shared board will appear directly as embedded applet in HTML pages as show in figure (6.19).

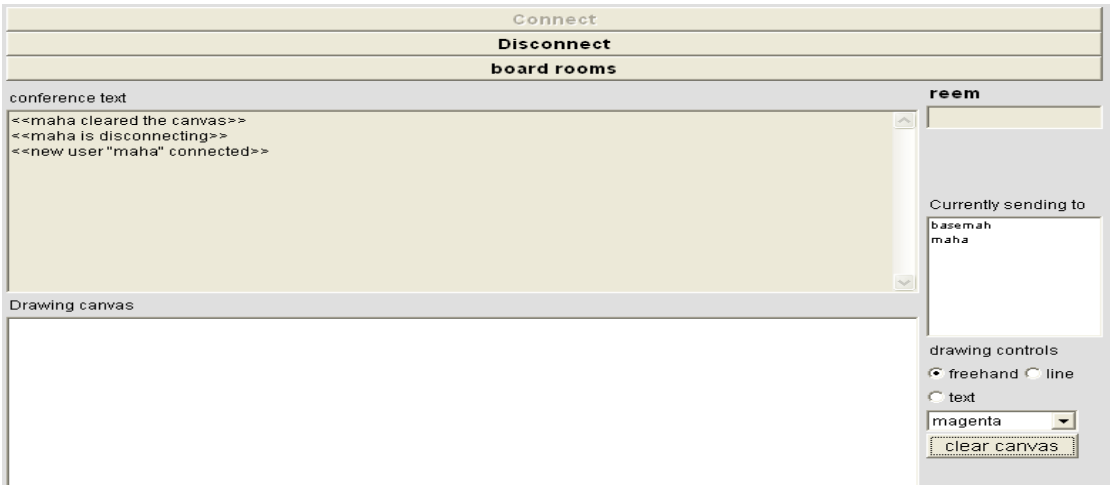



**Figure 6. 19:shared board**

The client can draw any thing to illustrates for author members of a group his/her idea as shown in figure (6.20).

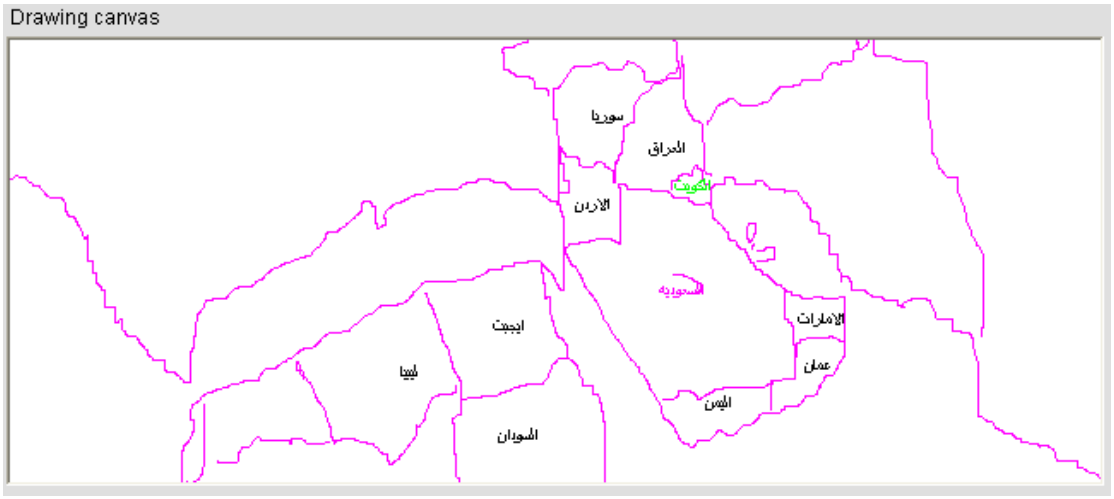



**Figure 6. 20:Drawing**

A user can change the color, the type of drawn even a line, text or a freehand and he/she can clean them figure (6.21).

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

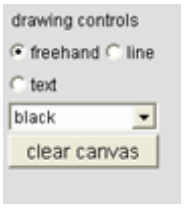



**Figure 6. 21:font color and type**

The shared board records all the events that happened as following:

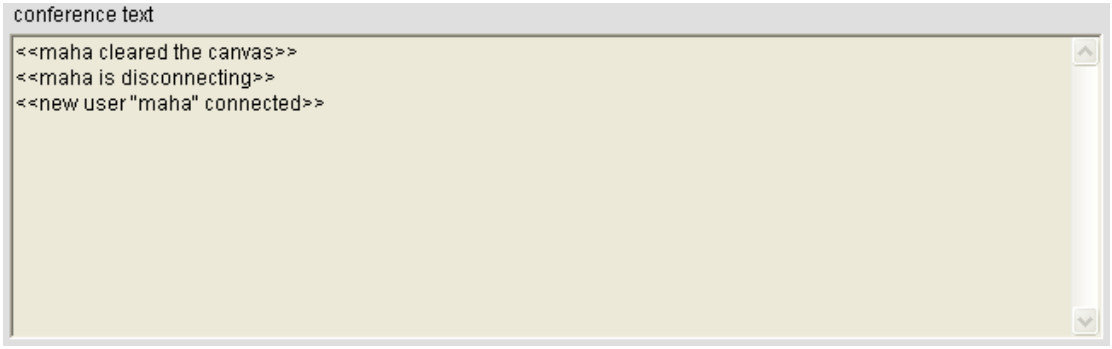



**Figure 6. 22:conference text.**

User can build a board room by pressing in the "create room" button in the following figure.

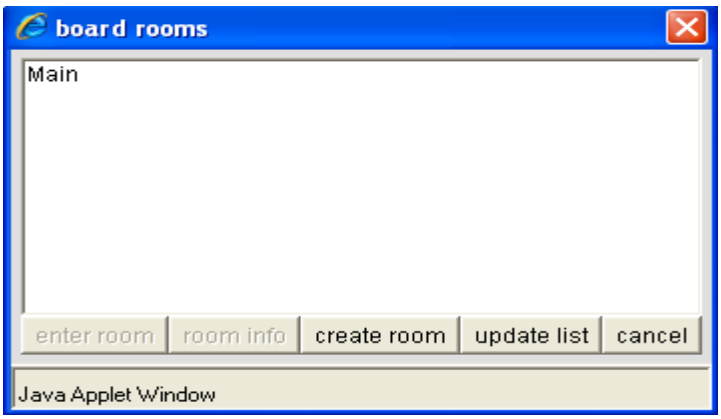



**Figure 6. 23:board rooms**

The user can let a room private by check on the box near of "is private" in the following figure (6.24).

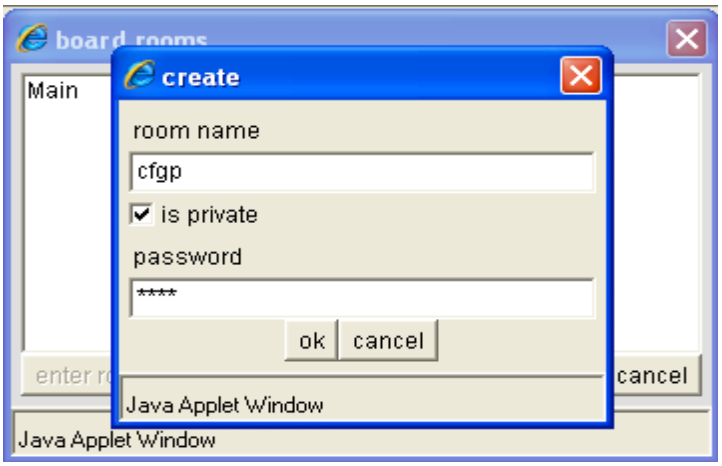



**Figure 6. 24: create room**

## 6.5 Discussion Room

There are two type of the discussion room as shown in figure (6.25)

**Figure 6. 25: Discussion room type.**

### 6.5.1. Public Discussion Room

Allow the supervisor to discuss with the group members.

### 6.5.2. Private Discussion Room

Just a coordinator and all other members of a group can enter this room.

The following figure (6.26) shows a screen shot from one chat room:

CFGP
Graduation project
LOGIN AS: User Name:maha | Type:member | Group Name:cfgp
Discussion Room
3 student(s) in this room
maha (member)
basemah (coordinator)
reem (member)
Welcome : maha , you are in room : cfgp
maha has joined.
basemah has joined.
basemah (10:01:16 ?) : alslam 3likum
maha (10:01:41 ?) : w3likum alsalam
basemah (10:02:24 ?) : we have meeting with D.Mona
maha (10:02:36 ?) : when?
basemah (10:03:04 ?) : next sunday in shaa allah
maha (10:03:24 ?) : ok..
reem has joined.
reem (10:04:33 ?) : what aboit report?
Send Message
Send
Refresh
Refresh
Log Out
Logout

**Figure 6. 26:Discussion Room.**

## 6.6 Shared Folder

When a member presses the following icon as shown in figure(6.27):

**Figure 6. 27:sharing folder icon.**

The next page will appear as shown in figure(6.28):

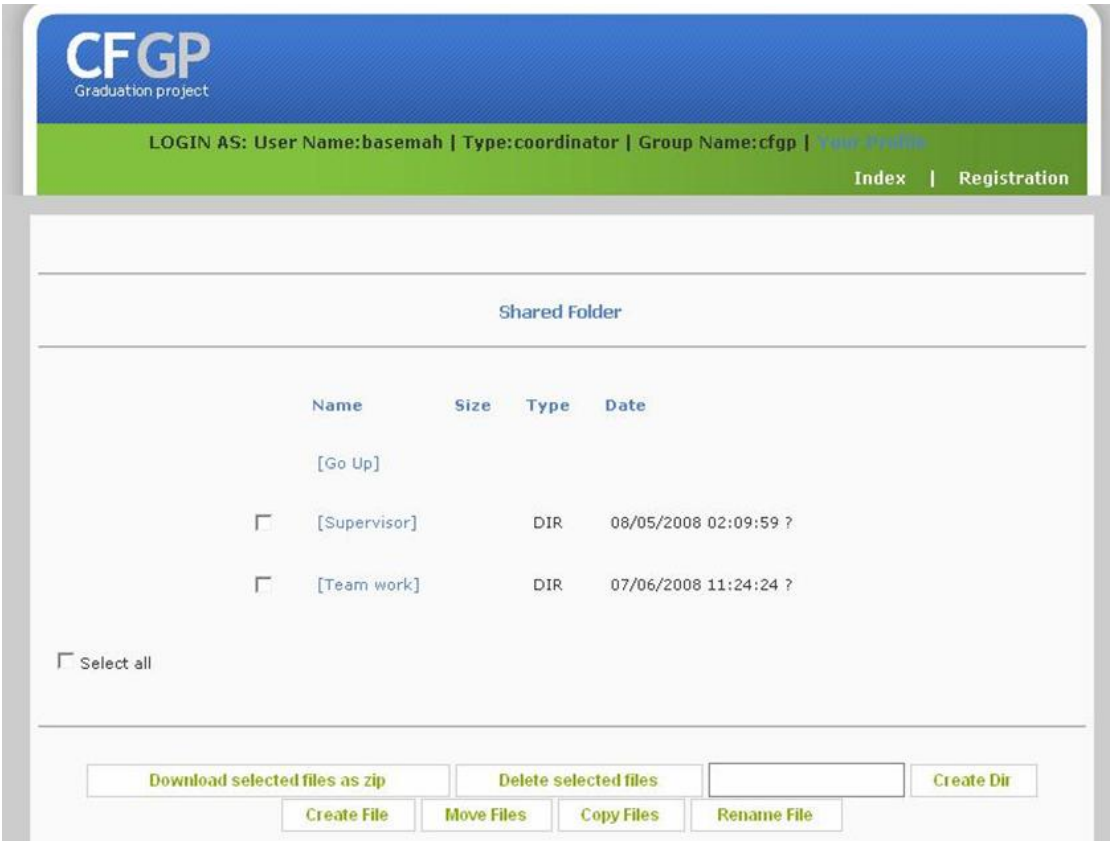



**Figure 6. 28:shared folder.**

The Figure (6.28) shows that there are two main folders which are "supervisor and Team work folder". The folders are represents between brackets […].

Figure (6.29) represent that inside the supervisor folder there are two folders "From and to ".

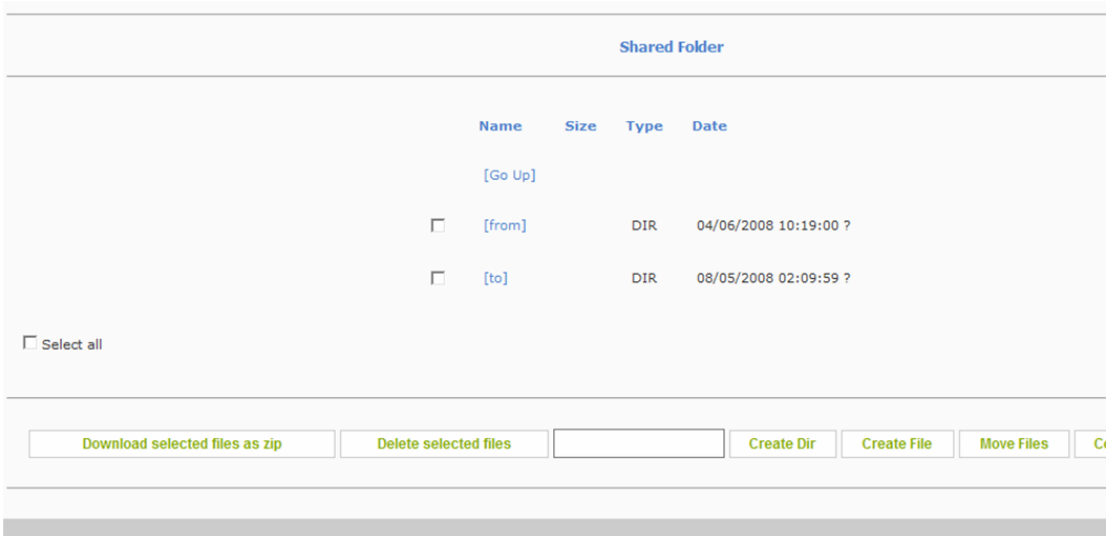



**Figure 6. 29:inside the supervisor folder.**

The files inside any folder supervisor folder or team work folder displayed by its name which is as a link allow the user to open it in a new page. Also, view the size, type and the date of creation or the last modification of the file. Moreover it contains two buttons "edit and downloads" . As shown in the following figure:

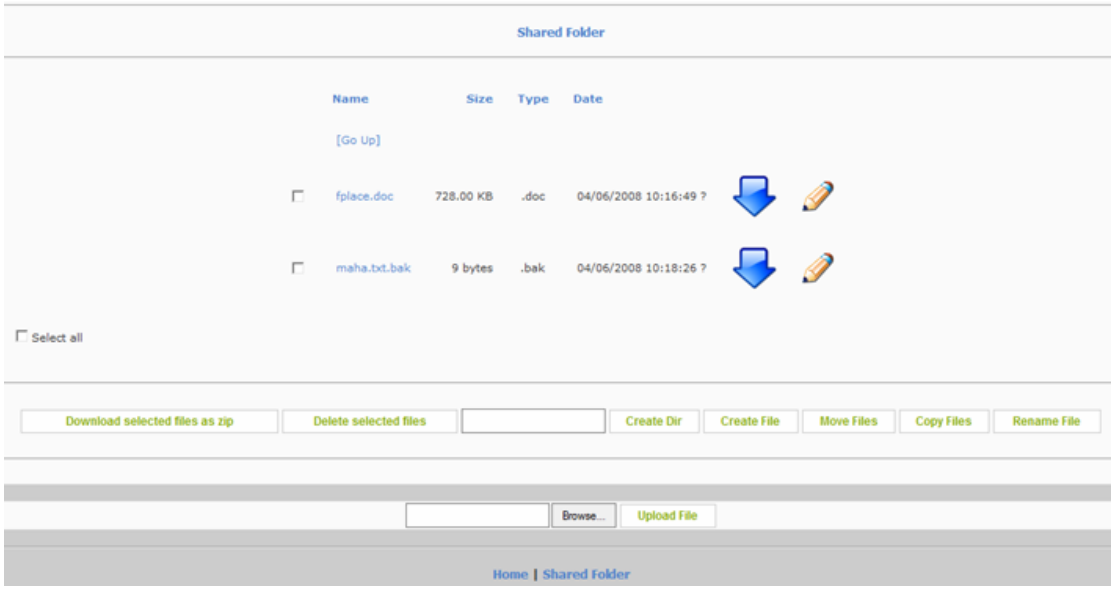



**Figure 6. 30:files display format**

If the user press downloads button the following figure (6.31) will appear:

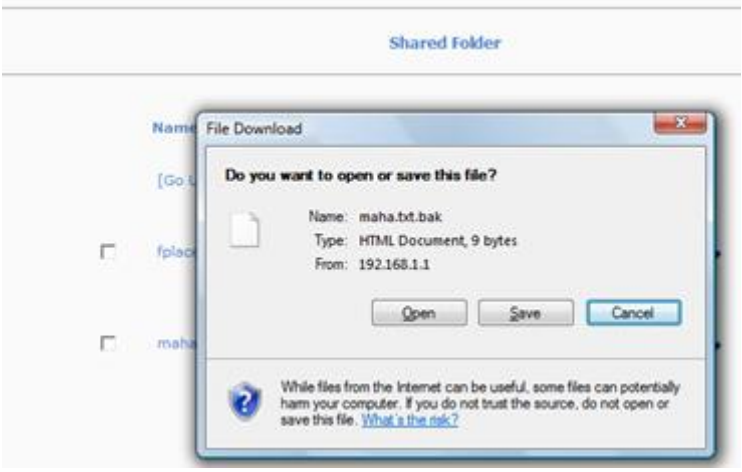

Shared Folder
File Download
Do you want to open or save this file?
Name: maha.txt.bak
Type: HTML Document, 9 bytes
From: 192.168.1.1
Open
Save
Cancel
While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

**Figure 6. 31:file download.**

And if the user presses edit button, figure (6.32) will appear:

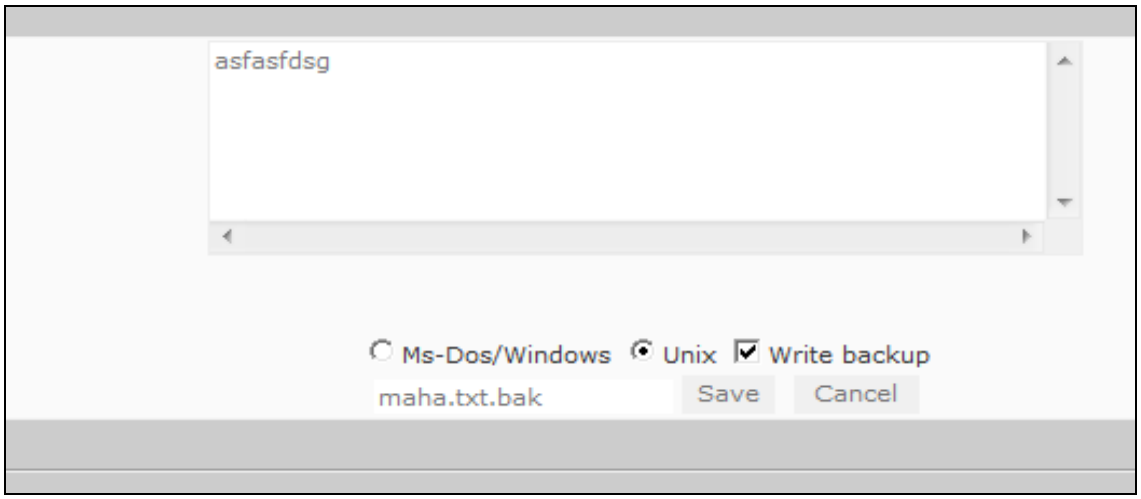



**Figure 6. 32:Edit file**

If a user try to open a folder or a file that he/she does not has the availability to open it figure (6.33) will display to him/her.

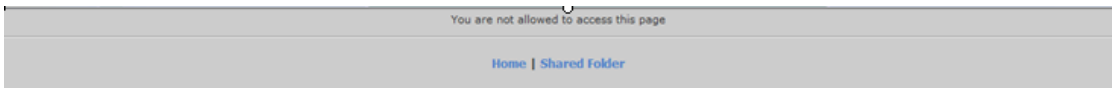



**Figure 6. 33: not allowed to access**

When a user wants to upload a file, he/she must press the browse button in the following figure (6.34).

**Figure 6. 34**: **upload file.**

Then, a following figure will appear to choose a file from his/her computer.

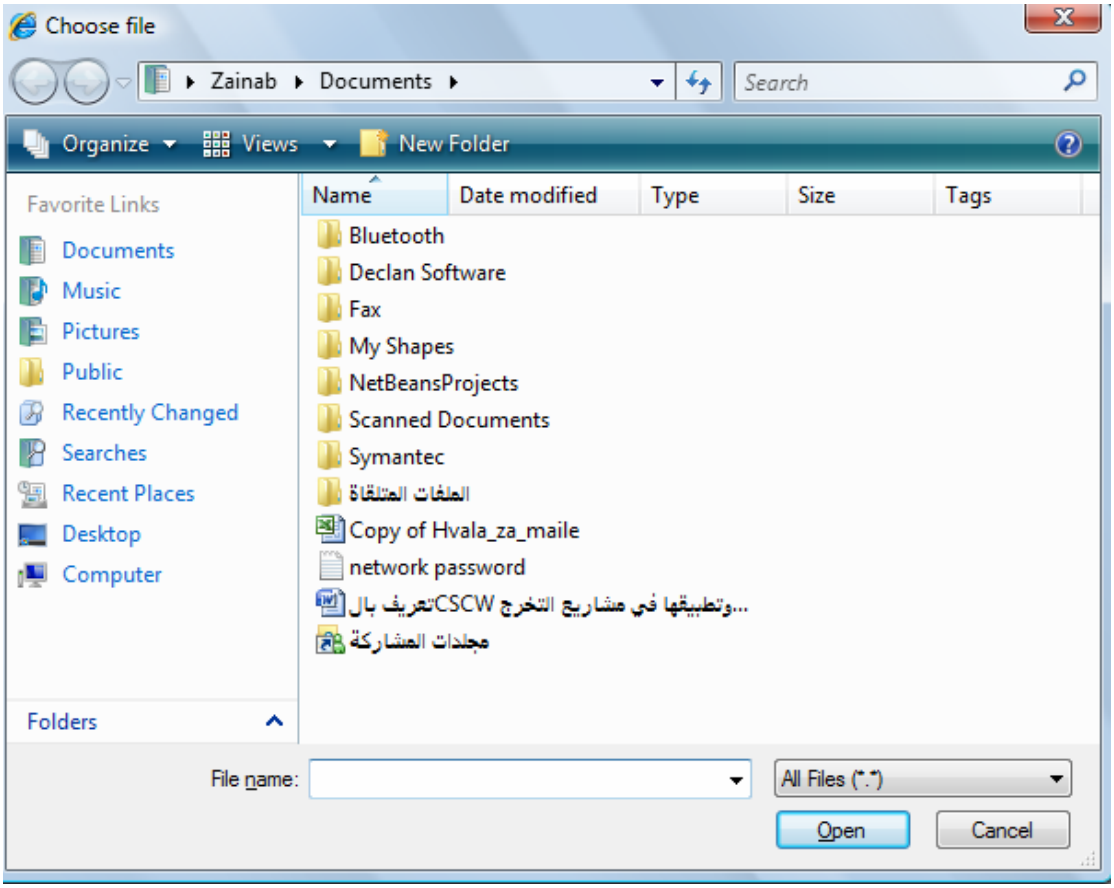



**Figure 6. 35: Choose file.**

When a user choose upload with out select a file figure (6.36) will display:

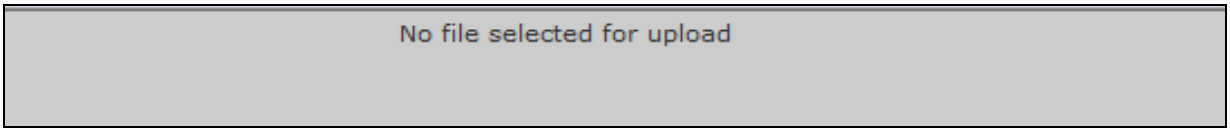



**Figure 6. 36: no selected file.**

If the user selects the file name, it will displayed in the text area as shown in the following:

**Figure 6. 37:file name**

After pressing upload button the file is begin download as in figure(6.38).

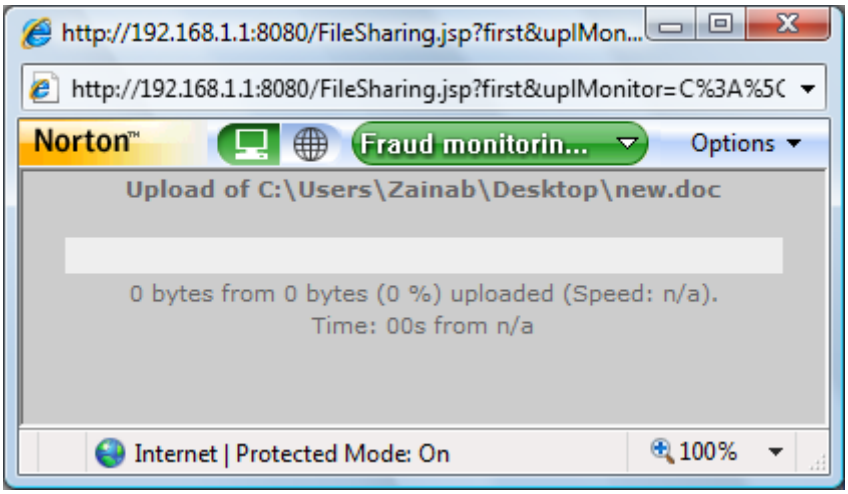



**Figure 6. 38:file is uploading**

If a user wants to delete a file, he/she selects file then presses delete selected file as following

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

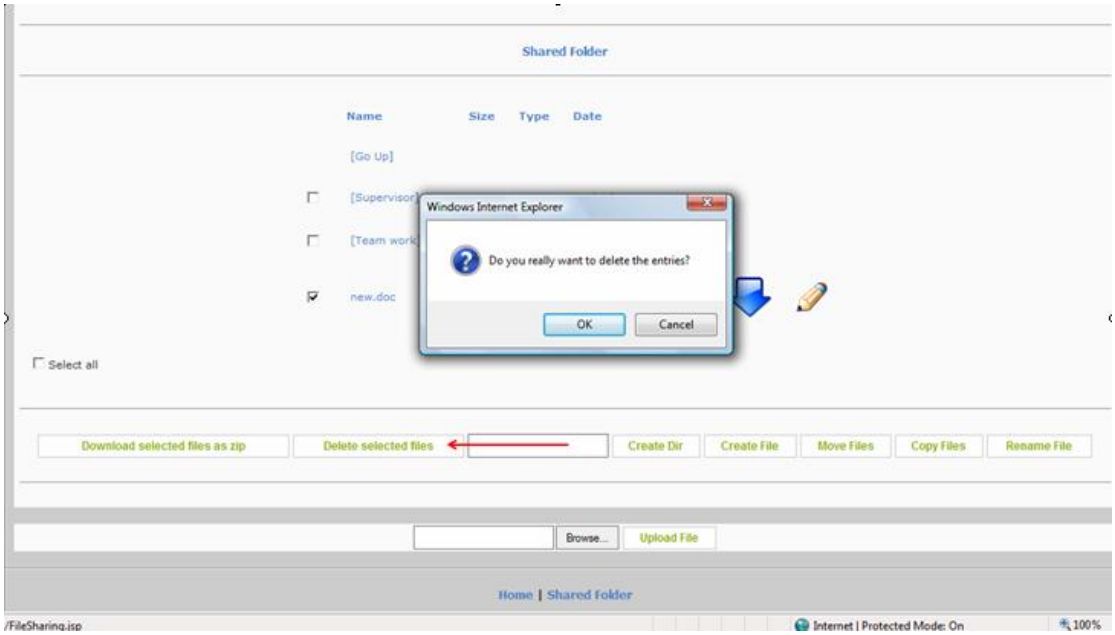



**Figure 6. 39:delete file.**

To create a new folder, write a folder name in the suitable text area then press create directory as following:

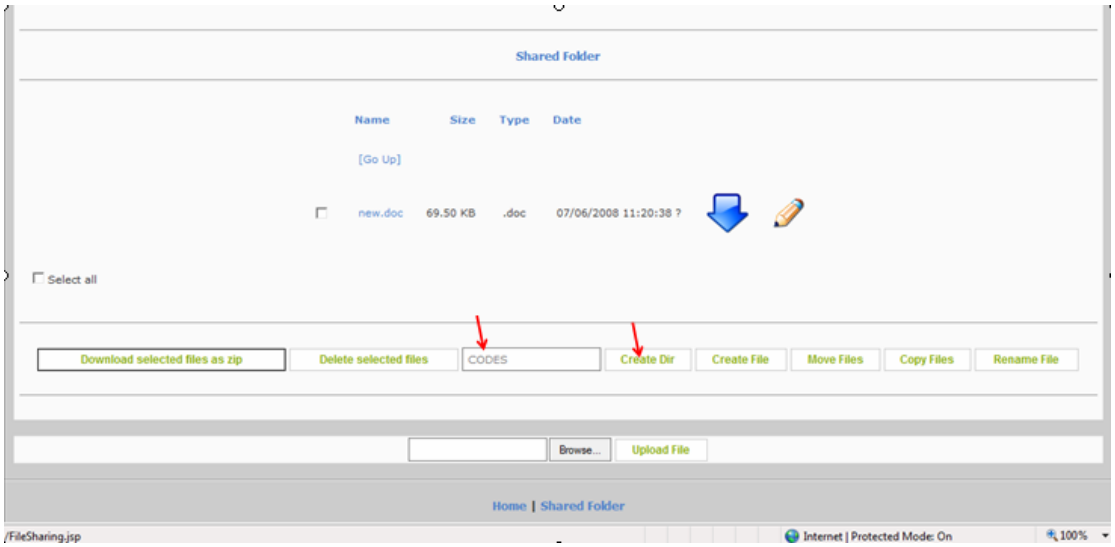



**Figure 6. 40:create new folder**

The figure(6.41) shows that folder is created.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

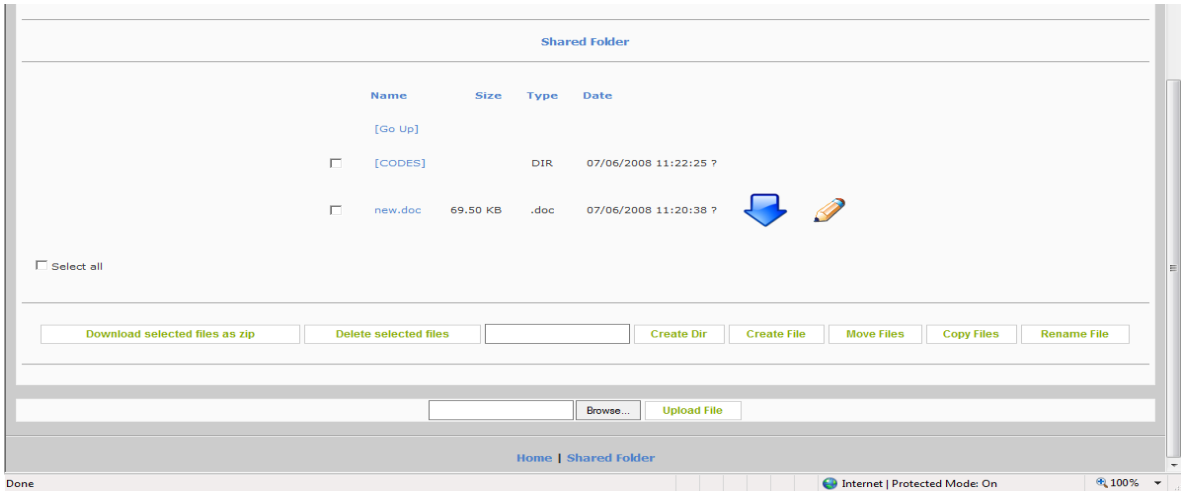



**Figure 6. 41:file created**

To create a new file, write a file name in the suitable text area then press create file as following:

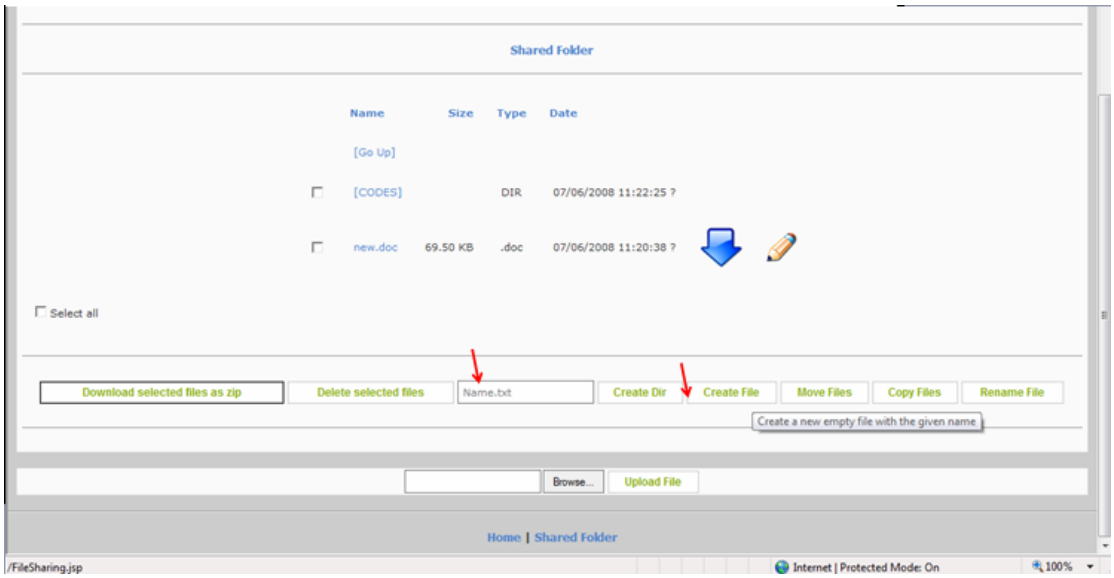



**Figure 6. 42:create a new file**

The figure(6.43) displays that file is created.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

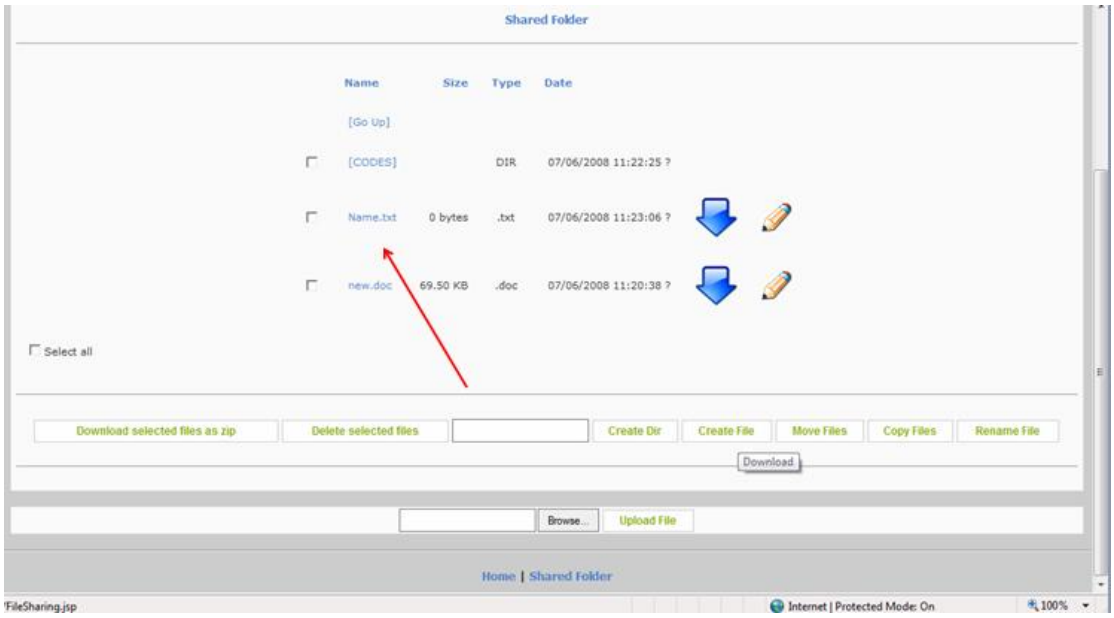



**Figure 6. 43**: **file is created**

To move files to another folder, a user must select the files which he/she wants to move it and writes a folder name in the text area then press move file button as shown in figure (6.44).

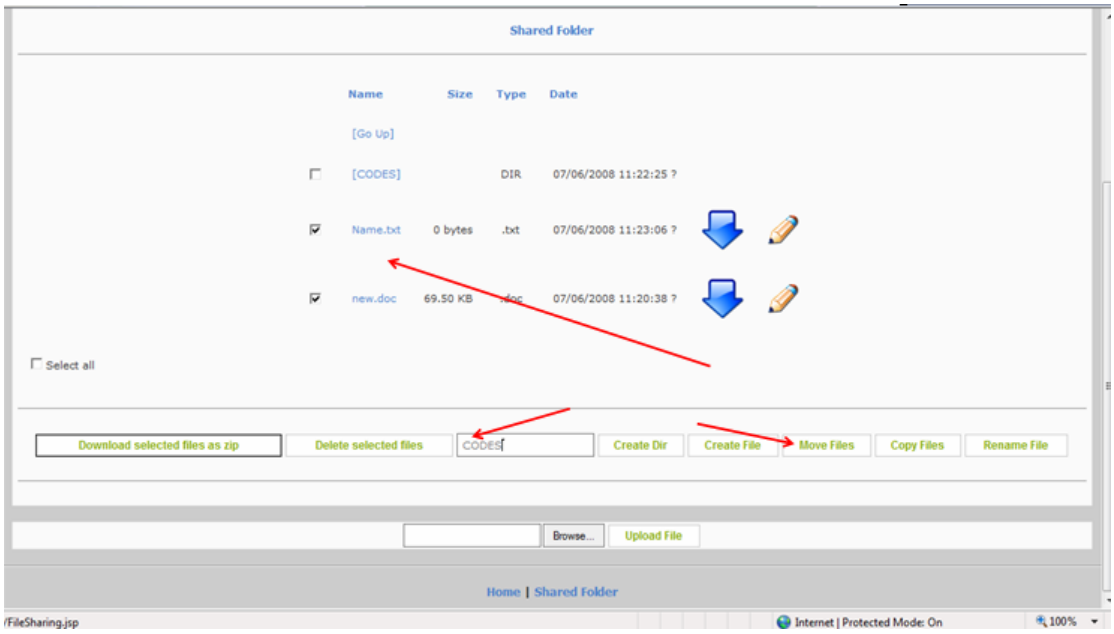



**Figure 6. 44**: **move file no another folder**

Figure (6.45) display that files are moved to the selected folder when a user point above a folder the name of seven files are shown as the following

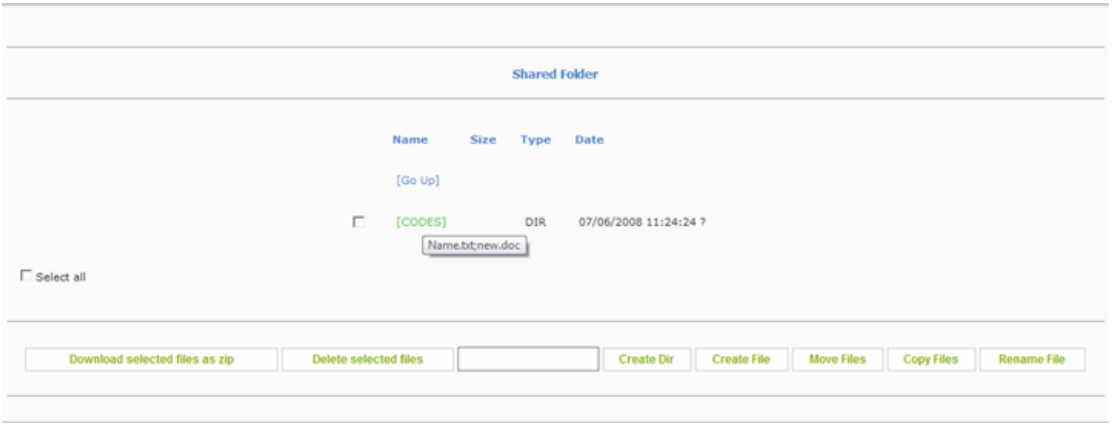



**Figure 6. 45:file moved**

To copy files to another folder, a user must select the files which he/she wants to copy it and write a folder name in the text area then press copy file button as shown in figure (6.46).

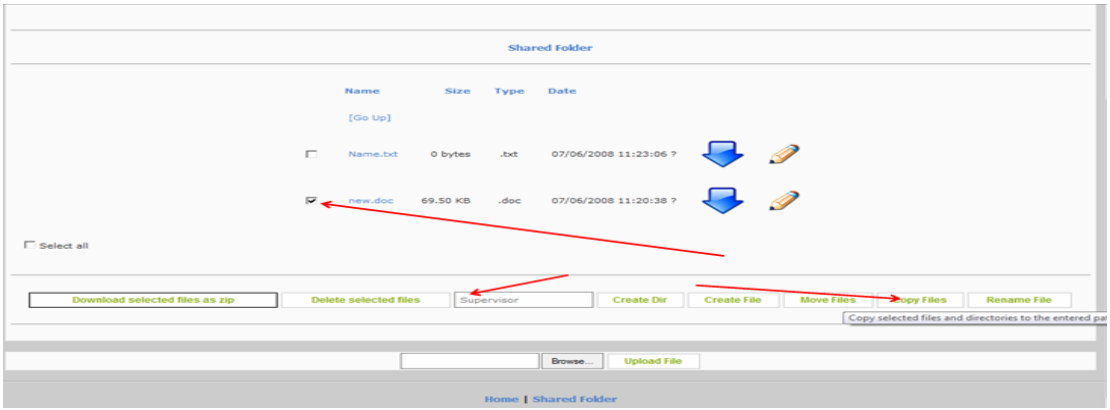



**Figure 6. 46:copy file to another folder.**

## 6.7 Voting module

Voting module allows team members to vote for the deadline meeting which is suggested by supervisor or coordinator.

### 6.7.1 Create a new meeting vote:

When a user want to create a new vote he/she must press the followin icon.

**Figure 6. 47:vote creation icon.**

This button Just appear in coordinator or a supervisor page because they are only who can add a vote for suggested meeting with all group members to discuss there work together in a discussion room in a specific suggested date and time as shown in figure(6.48).

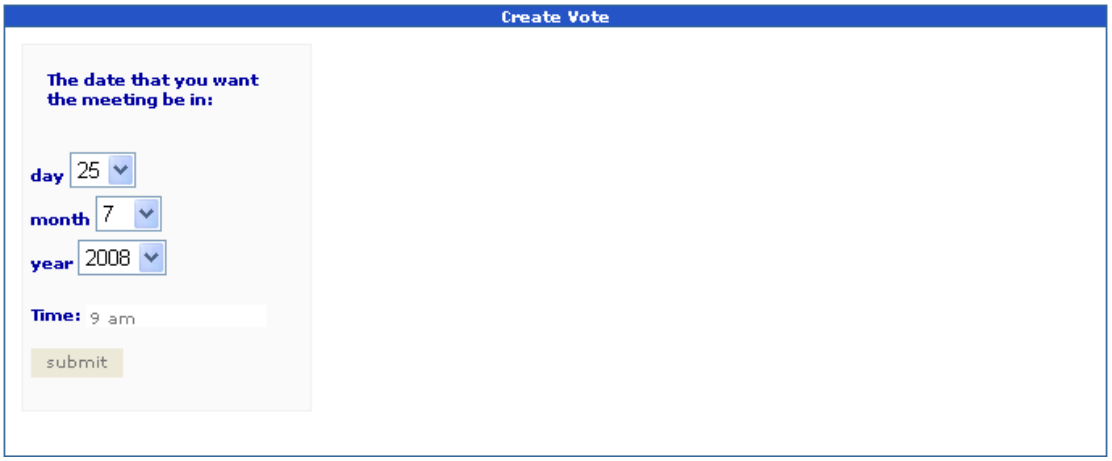



**Figure 6. 48:create vote.**

### 6.7.2. View Vote

After pressing submit, a meeting vote form will appear in the all members pages in figure (6.49). They should vote to determine if they are agreeing of that date and time for meeting or not.

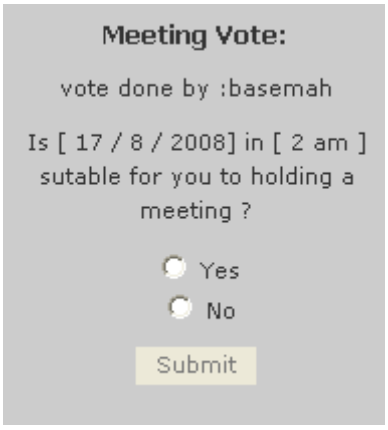



**Figure 6. 49:view vote**

### 6.7.3 Voting Result

The result of voting appaer to the member as shown in figure (6.50) and according to this result determine the date of meeting.

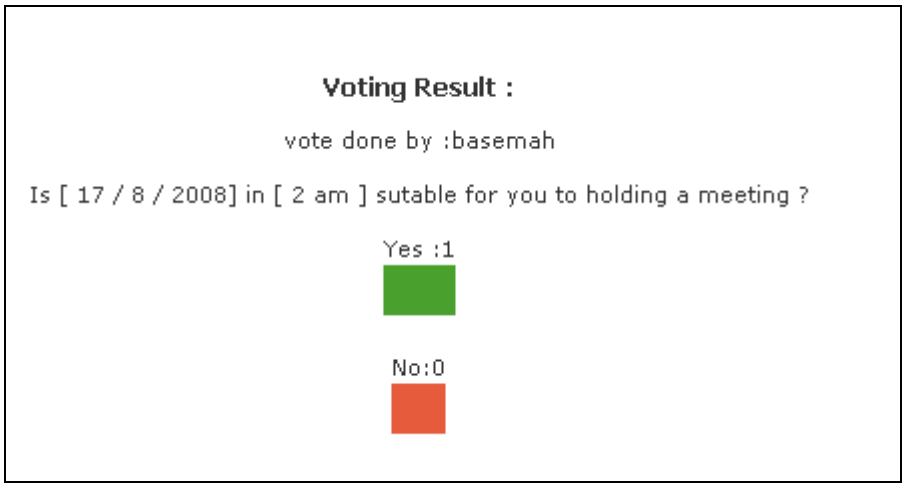



**Figure 6. 50:voting result.**

Before one day of the meeting, date which are determined a system will display if there is a meeting or not as shown below.

**Figure 6. 51:meeting date.**

if there is no meeting the figure (6.52) will appear

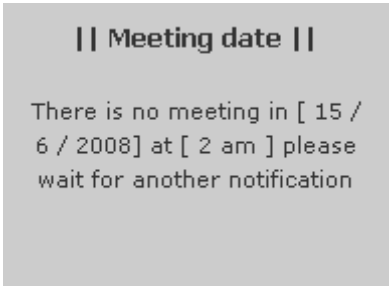



**Figure 6. 52: no meeting**

## 6.8. Member Profile

A member of a group can change his/her profile, choose a picture or change his/her E-mail and customize his/her signature as shown in figure (6.53).

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

**Figure 6. 53:member profile.**

## 6.9 Advertising Space

The advertising space will appear by press the advertising space icon as shown in figure (6.54).

**Figure 6. 54:advertising space icon**

after press figure(6.55) all advertisments sections will appear to the member as following.

Advertising Space

| Section | Topics | Replies | Last Post Info | Delete Section |
|---|---|---|---|---|
| Achievements | 1 | 0 | 2008-05-01 10:44:08.0<br>In: voting<br>By: basemah | Delete |
| Meeting<br>with Supervisor. | 4 | 0 | 2008-05-01 22:16:19.0<br>In: Meeting in 15/5/1429<br>By: System | Delete |
| ToDo | 2 | 1 | 2008-04-30 07:38:17.0<br>In: to complete#2<br>By: basemah | Delete |

**Figure 6. 55:advertising space**

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

Only the coordinater can add sectoin of advertisments as shown in figure(6.56), each section has advertisments which is related to the same subject.

**Figure 6. 56:add section**

Any member of a group can add an advertisement in the related section to attract member's attention to the subject he/she want as shown in figure(6.57).

**Figure 6. 57:add an advertisement**

In the following figure (6.58), member of a group can reply of an advertisement with his/her opinion.

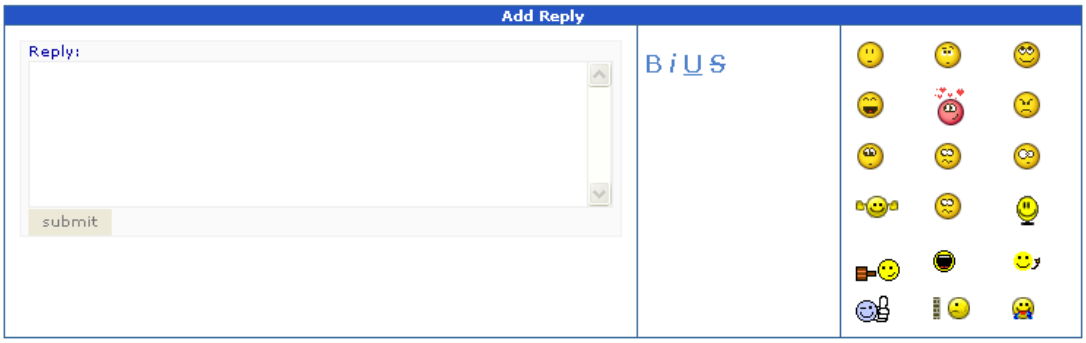



**Figure 6. 58:add replay**

## 6.10 Task Schedule

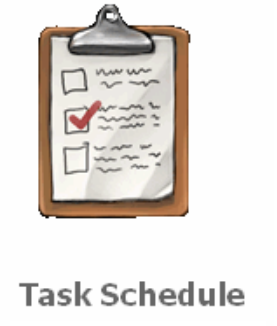



**Figure 6. 59:Task schedule icon**

After member or coordinator pressing the above button the tasks schedule, which is scheduled by the coordinator will be displayed as following.

| Task | Day | Month | Year | Time | Edit | Delete |
|---|---|---|---|---|---|---|
| planing in paper | 10 | May | 2008 | 10:30 | | |
| write chapter introduction in report | 11 | May | 2008 | 5 pm | | |
| write source code | 9 | June | 2008 | 1:50 | | |
| please comlete the report | 12 | June | 2008 | 10 am | | |

**Figure 6. 60: Task schedule.**

The coordinator can add, edit and delete task.

### 6.10.1 Add task

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

The coordinator must determine the task, date and time that must be finish in it, see figure (6.61).

.

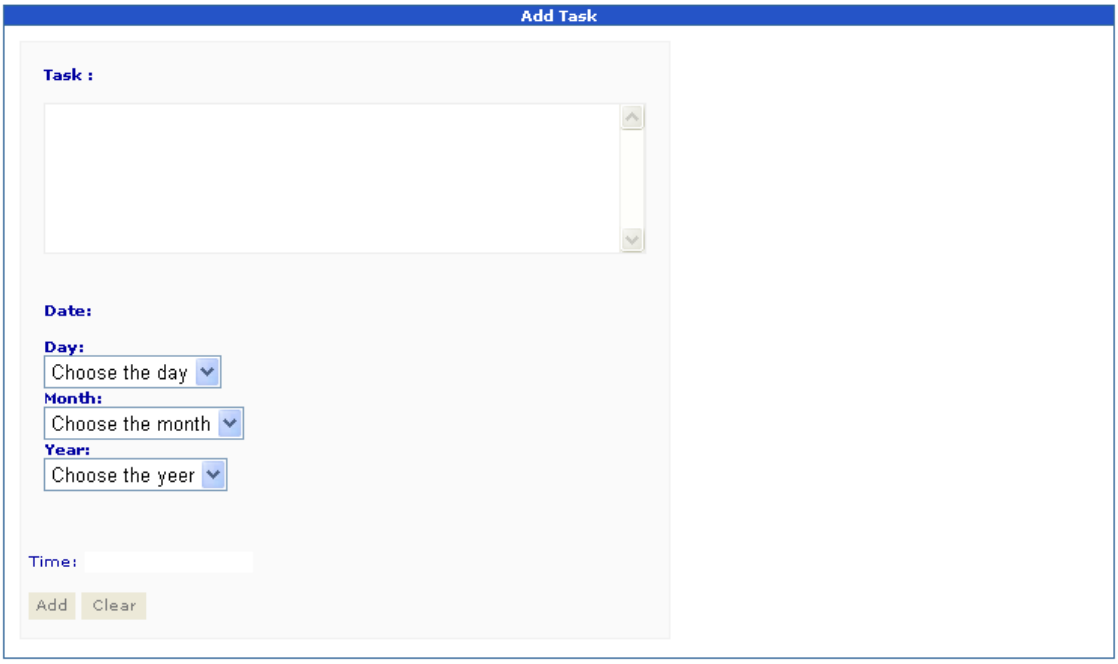



**Figure 6. 61: add task**

## 6.10.2 Edit task

The coordinator can change the task, date and time that must be finish in it as following.

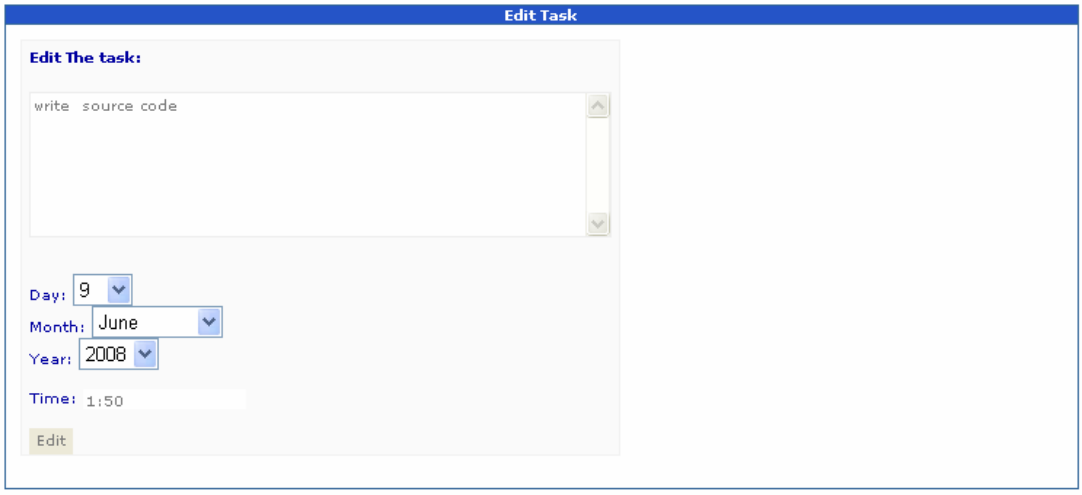



**Figure 6. 62: Edit task**

## 6.11 Log File

**Figure 6. 63: log file icon**

The CFGP web based tool after pressing the above button displays the last ten events with time and members name that happened.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

| Event | Date |
|---|---|
| may: Add Rerlay to This Advertisment(click here) | 2008-05-02 05:00:12 |
| zainab: Add Rerlay to This Advertisment(click here) | 2008-05-02 04:59:15 |
| reem: Add Rerlay to This Advertisment(click here) | 2008-05-02 04:52:46 |
| sara: Add Rerlay to This Advertisment(click here) | 2008-05-02 04:50:45 |
| basemah: Add vote which is( Is [ 1/5/1429 ] in [ 5 pm ] sutable for you to holding a meeting ? | 2008-05-02 04:35:25 |
| basemah: Add Advertisment to This Section (click here) | 2008-05-02 04:27:04 |
| basemah: Edit Task to: [ write source code In ( 9 / June / 2008 ) ,Time is: 1:50 | 2008-05-02 04:19:37 |
| basemah: Delete Task | 2008-05-02 04:14:37 |
| basemah: Add Task which is [ please comlete the report In ( 12 / June / 2008 ) ,Time is: 10 am | 2008-05-02 04:07:32 |
| maha: Add Rerlay to This Advertisment(click here) | 2008-05-02 03:51:40 |

**Figure 6. 64:last 10 event**

## 6.12 Delete a group

Any group of CFGP web based tool can deleted by group's coordinator as shown in figure (6.65).

**Figure 6. 65: Delete group.**

# 7. Conclusion

Collective actions of graduation project require a great deal of interaction between members of the team and communicate directly to complete the tasks of this project. While team members are presented in different places with the difficulty of direct communication a useful work environment needed to provide communication services that are required to completion of the project.

The topic of Computer Supported Cooperative Work (CSCW) has attracted much attention in the last few years. CSCW is identifiable research field focused on

understanding characteristics of interdependent group work with the objective of designing adequate computer-based technology to support such cooperative work.

CSCW goes through many application areas, in CFGP we set out an approach to CSCW as a field of academic collaborative work which we believe provides a coherent conceptual framework for this area. CFGP designed to give an effective support for the dynamic control of the co-operative work in graduation project.

CFGP web based tool attempts to be an appropriate environment for accomplishment graduation project by considering required needs as team of supervisor, coordinator and members. This shown clearly by internal design of CFGP features which are shared folders ,shared board, private and public discussion rooms, advertisement board ,vote and schedule tasks. Moreover that shown by implemented CFGP web based tool as web site for accessibility and portability.

In addition to technical challenges, the starting point has substantially been important challenge. CSCW needs to deeply study in order to get appropriate tool for the intended or actual users. As a groupware tool CFGP requires more efficient technologies of communications and reusing knowledge in design process.

CFGP web based tool is based on client-server. Java programming was best choice to achieving powerful tool in terms of flexibility, security, ease of use and extensibility; so, all CFGP components have been implemented with servlet/jsp. Database is main issue of CFGP web based tool, MySql was appropriate choice to implement CFGP database for many reasons such as scalability and flexibility, availability, web and data warehouse strengths, etc.

Either for improving reason or flexibility of graduation project domain and needs, CFGP web based tool could be improved in many different ways. One of improvements could be made is to improve the collaboration work by adding editor that support multi-editors. Make optional discussion room for sharing discussions with other group in concerned issues and ability to open private and public room in parallel. Improve CFGP security by applying more efficient cryptography algorithm

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

on more than passwords such as instances messages in chatting, transfer files, etc. Also an additional validity to members and supervisors can be added by the coordinator manually from the control panel of the tool. Other gadgets like calendars, clocks, sending emails will be suitable for join to the collections of gadgets CFGP have.

# Bibliography

[1] Irene. G. , Computer-Supported Cooperative Work , Morgan Kaufmann , 1988.

[2] Kerr E.B. and Hiltz S.R. , Computer Mediated Communication Systems - Status and Evaluation ,Academic Press, NewYork , 1982.

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

[3] Pullinger D.J. , Chit-Chat to Electronic Journals: Computer Conferencing Supports Scientific Communication IEEE trans, on Professional Communications, Vol PC 29, No 1,pp23-29, March 1986.

[4] Lauwers J.C. and Lantz K.A. , Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems Proceedings of CHI 90 Seattle, Washington April1-5, 1990, ACM press, ISBN- 0-201-50932-6.

[5] Kraemer K. L. and Kling J. L. , Computer based systems for cooperative work and group decision making , ACM Computing Surveys, Vol 20, No 2, June 1988.

[6] Hahn U., et al. , Coauthor: A hypermedia Group Authoring Environment In Proceeding of EC-CSCW, the 1st European Conference on CSCW.

[7] Atul A. , Jon H. , Marvin T. ,William J. and John R. , Cooperative Task Management without Manual Stack Management , In: USENIX Annual Technical Conference, General Track , 2002.

[8] L. Navarro , W. Prinz and T. Rodden Towards Open CSCW Systems ,In Proc. IEEE 3rd Workshop on Future Trends of Distributed Computing Systems, Taipei, Taiwan, April. 1992.

[9] Dean G., Hutchison D., Rodden T. and Sommerville I. , Cooperation and configuration within distributed systems management , Configurable Distributed Systems, 1992., International Workshop on Volume , Issue , 25-27 Mar 1992 Page(s):274 - 285 Digital Object Identifier.

[10] Johansen, R., Groupware: Computer Support for Business Teams , The Free Press, N.Y., 1988

[11] J.A. Mariani and T.A. Rodden , The Impact of CSCW on Database Tecnology , Department on computer science, Lancaster , LA1 4YR United kingdom,Berlin,Germany,1991

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

[12] Wooldridge M. and Jennings, N. Intelligent agents: theory and practice. The Knowledge Engineering Review , 1995.

[13] Goscinski A. Distributed Operating Systems , The logical design, Addison Wesley, Sydney, 1991.

[14] LYNCH , N.A., FISCHER and M.J. On describing the behavior and implementation of distributed systems In Theoretical Computer science , North-Holland, Amsterdam, 1981.

[15] Jones , M.B., and R.F. Rashid , Mach and Matchmaker: Kernel and Language Support for Object-Oriented Distributed Systems, Portland, Oregon, 1986.

[16] Hopper, A., Pandora - An Experimental System for Multimedia Applications, ACM Operating Systems Review, April 1990.

[17] Blair ,et al. , Incorporating Multimedia in Distributed Open Systems , Norway, May 1991.

[18] T. Rodden and G.Blair , Distributed Systems Support for Computer Supported Cooperative Work ,U.K: Lancaster University. [E-book] Available: citeseer e-book.

[19] ] D.Twitchell, M.Adkins, J.Nunamaker and J.Burgoon. Using Speech Act Theory to Model Conversations for Automated Classification and Retrieval . USA: University of Arizona,2004. [E-book] Available:

[20] T. Rodden . A Survey of CSCW Systems . Bailrigg: Lancaster University. [E-book] Available:http://www.sesis.com.br/publicacoes/pdfs/rodden92survey.pdf .

[21] ”Computer Supported Cooperative Work (CSCW) ,” Springer Link, April , 2002,

Available:http://www.springerlink.com/content/0925-9724 .10

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

[22] ”IPsec Protocol Suite, ” October 2005 . [Online]. available: http://docs.hp.com/en/J4256-90015/ch01s02.html .

[23] TimBL , ”Basic HTTP as defined in 1992 ,” 1992 . [Online]. Available:

http://www.w3.org/Protocols/HTTP/HTTP2.html.

[24] JavaBeat, ”Java Server Pages(JSP),” Java Beat ,2006 [Online]. Available: http://jsp.javabeat.net/.[Accessed: March. 26, 2008].

[25] MySQL AB, ”MySQL,” 1995-2008. [Online]. Available: http://www.mysql.com/. [Accessed: March. 26,2008].

[26] T.Ylonen ”The SSH(Secure Shell)Remote Login Protocol,” 1995-1996 [Online].Available:http://www.free.lp.se/fish/rfc.txt [Accessed: June. 5, 2008].

[27] ] Kipp E.B.Hickman ”The SSL Protocol ,” 1994 [Online].Available:

http://www.webstart.com/jed/papers/HRM/references/ssl.html [Accessed: June. 5, 2008].

[28] Marty H and Larry B , Core Servlets and JavaServer Pages ,Edition 2, Vol.(1).Prentice Hall PTR,2006.

[29] 2008Sun Microsystems ”Java SE Security,” 1994-2008[Online].Available:

http://java.sun.com/products/jce/

[30] ”Sun Java Crypto API ,” [Online].Available: http://java.sun.com/j2se/1.4.2/docs/api/javax/crypto/packagetree.html[Accessed: May. 30, 2008].

[31] ”E-Learning Computer Security,” [Online].Available: http://webdev.sis.pitt.edu/st/crypto4.htm[Accessed:May. 30, 2008]

[32] JavaTM 2 Platform,”[Online].Available:http://java.sun.com/j2se/1.3/docs/api/

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

[Accessed:May.24,2008].

[33] Jeffrey M. Hunter.”Java Networking Programming Overview,” 2008.[Online]. Available:http://www.idevelopment.info/data/Programming/java/networking/JavaNetworkingProgrammingOverview.html.[Accessed: May 30, 2008].

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية

مها الفوزان

قسم مهارات تطوير الذات

عمادة السنة التحضيرية